

[illegible][illegible]

```
PPPPPPPP      AAAAAA      IIIIII      NN      NN      TTTTTTTTTT      RRRRRRRR
PPPPPPPP      AAAAAA      IIIIII      NN      NN      TTTTTTTTTT      RRRRRRRR
PP      PP      AA      AA      II      NN      NN      TT      RR      RR
PP      PP      AA      AA      II      NN      NN      TT      RR      RR
PP      PP      AA      AA      II      NNNN      NN      TT      RR      RR
PP      PP      AA      AA      II      NNNN      NN      TT      RR      RR
PPPPPPPP      AA      AA      II      NN      NN      TT      RRRRRRRR
PPPPPPPP      AA      AA      II      NN      NN      TT      RRRRRRRR
PP      AAAAAAAAAA      II      NN      NNNN      TT      RR      RR
PP      AAAAAAAAAA      II      NN      NNNN      TT      RR      RR
PP      AA      AA      II      NN      NN      TT      RR      RR
PP      AA      AA      II      NN      NN      TT      RR      RR
PP      AA      AA      IIIIII      NN      NN      TT      RR      RR
PP      AA      AA      IIIIII      NN      NN      TT      RR      RR
```

```
LL      IIIIII      SSSSSSSS
LL      IIIIII      SSSSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SSSSSS
LL      II      SSSSSS
LL      II      SS
LL      -II      SS
LL      II      SS
LL      II      SS
LLLLLLLLLLLL      IIIIII      SSSSSSSS
LLLLLLLLLLLL      IIIIII      SSSSSSSS
```


(2)	263	DEFINITIONS
(3)	297	PPD SEND ROUTINES
(3)	298	- SNDDG SEND DATAGRAM
(4)	362	- SNDMSG SEND SEQUENCED MESSAGE
(5)	440	- REQDAT REQUEST BLOCK DATA
(6)	469	- SNDDAT SEND BLOCK DATA
(7)	496	- READCNT READ COUNTERS
(8)	539	- MRESET MAINTENANCE RESET
(9)	577	- MSTART SEND MAINTENANCE START
(10)	616	- CLRCACHE, CLEAR ANY PPD LAYER CACHES
(11)	673	PA INTERRUPT_SERVICE ROUTINE
(12)	912	HANDLE INT, HANDLE PORT INTERRUPT
(13)	1040	HANDLERS FOR RESPONSES WITH GOOD STATUS
(13)	1041	- REC_CNFRFC, SEND DATA IS COMPLETE
(13)	1042	- REC_DATREC, REQUEST DATA IS COMPLETE
(14)	1082	- REC_DGREC, PROCESS RECEIVED DG
(15)	1129	- REC_IDREC, PROCESS RECEIVED ID
(16)	1166	- REC_LBREC, PROCESS RECEIVED LB DG
(17)	1192	- REC_MSGREC, PROCESS RECEIVED MESSAGE
(18)	1233	- REC_RDCNT, PORT COUNTERS READ
(19)	1266	- REC_SETCKT, SET CIRCUIT DONE
(20)	1317	- REC_SNDDG, PROCESS SENT DATAGRAM
(20)	1318	- INT\$DISP SENDDG, DISPATCH A SENT DG
(21)	1373	- REC_SNDMSG, PROCESS SENT MESSAGE
(22)	1406	- REC_REQID, ID REQUESTED
(22)	1407	- REC_SNDRST, RESET SENT
(22)	1408	- REC_SNDSTRT, START SENT
(23)	1442	PROCESSING OF ERROR STATUS IN RESPONSE
(24)	1540	- MACROS TO DEFINE ACTION TABLE
(25)	1662	- OPCODE-DEPENDENT ERROR ACTION TABLE
(26)	1760	- RSP_ERROR, DISPATCH ON ERROR
(26)	1761	- TYPE
(27)	1817	- RSP_PATH_FAIL, PROCESS SINGLE PATH
(27)	1818	- FAILURE
(28)	1916	- RSP_UNREC_PKT, PROCESS RECEIPT OF
(28)	1917	- UNRECOGNIZED PKT
(29)	1975	- RSP_NO_PATH, PROCESS NO PATH
(29)	1976	- STATUS
(29)	1977	- RSP_PKTSIZ_VIO, PROCESS PACKET SIZE
(29)	1978	- VIOLATION STATUS
(29)	1979	- RSP_VC_CLOSED, PROCESS VC CLOSED
(29)	1980	- STATUS
(30)	2068	ACTION ROUTINES
(30)	2069	- RSP_CACHECLR
(31)	2114	- RSP_CLOSED_VC
(32)	2144	- RSP_CRASH_PORT
(33)	2177	- RSP_CRASH_VC
(34)	2207	- RSP_DRAIN_ERR
(35)	2251	- RSP_IGNORE_ERR
(35)	2252	- RSP_DISCARD_ERR
(36)	2287	- OPTIONAL DEBUG BUGCHECKS
(37)	2373	- INT\$FATALQ_IDFQ, ERROR INSERTING ON DFQ
(37)	2374	- INT\$FATALQ_IMFQ, ERROR INSERTING ON MFQ
(37)	2375	- INT\$FATALQ_CQL, ERROR INSERTING ON COMQL
(37)	2376	- INT\$FATALQ_CQH, ERROR INSERTING ON COMQH
(37)	2377	- INT\$FATALQ_RSPQ, ERROR REMOVING FROM RSPQ
(37)	2378	- INT\$FATALQ_RDFQ, ERROR REMOVING FROM DFQ
(37)	2379	- INT\$FATALQ_RMFQ, ERROR REMOVING FROM MFQ
(38)	2463	PACKET ALLOCATION/DEALLOCATION/DISPOSAL ROUTINES
(38)	2464	- INT\$INS_FREEQ, DETERMINE IF PKT

(38)	2465	-		IS MSG OR DG AND
(38)	2466	-		INSERT OF FREE QUEUE
(39)	2492	-	INS_MFREEQ	INSERT ON MESSAGE FREE QUEUE
(40)	2529	-	INS_DFREQ	INSERT ON DATAGRAM FREE QUEUE
(41)	2580	-	INT\$ALLOC_MSG,	ALLOCATE A MSG BUFFER FROM POOL
(41)	2581	-	INT\$ALLOC_DG,	ALLOCATE A DG BUFFER FROM POOL
(41)	2582	-	INT\$ALLOC_DGPPD,	ALLOCATE A BUFFER FOR
(41)	2583	-		PPD COMMAND
(42)	2701	-	INT\$DEAL_MSG,	DEALLOCATE A MESSAGE BUFFER
(42)	2702	-	INT\$DEAL_DG,	DEALLOCATE A DATAGRAM BUFFER
(42)	2703	-	INT\$DEAL_PKT,	DEALLOCATE A DG OR MSG
(43)	2770	-	DFQ2POOL	REMOVE FROM DATAGRAM FREE QUEUE
(44)	2790	-	MFQ2POOL	REMOVE FROM MESSAGE FREE QUEUE


```

0000 1      .TITLE  PAINTR
0000 2      .IDENT  'V04-001'
0000 3
0000 4      *****
0000 5      *
0000 6      *  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 7      *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 8      *  ALL RIGHTS RESERVED.
0000 9      *
0000 10     *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 11     *  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 12     *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 13     *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 14     *  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 15     *  TRANSFERRED.
0000 16     *
0000 17     *  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 18     *  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 19     *  CORPORATION.
0000 20     *
0000 21     *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 22     *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 23     *
0000 24     *
0000 25     *****
0000 26
0000 27     ++
0000 28
0000 29     FACILITY:
0000 30
0000 31         VAX/VMS EXECUTIVE, I/O DRIVERS
0000 32
0000 33     ABSTRACT:  CI PORT INTERRUPT SERVICE AND RESPONSE DISPATCHING
0000 34
0000 35     AUTHOR:  N. KRONENBERG,  MAY 1981
0000 36
0000 37     MODIFIED BY:
0000 38
0000 39         V04-001 NPK3066      N. Kronenberg      7-Sep-1984
0000 40         Upon receipt of an unrecognizable packet, use
0000 41         INT$ALLOC PPDDG rather than INT$ALLOC_DG (which returns
0000 42         offset within buffer to application data) to get a
0000 43         buffer in which to command the port to inhibit dg
0000 44         reception from the incoherent port.
0000 45
0000 46         V03-027 NPK3065      N. Kronenberg      23-Aug-1984
0000 47         Fix a .BSBW needed for queue checking and/or pkt
0000 48         tracing.
0000 49
0000 50         V03-026 NPK3058      N. Kronenberg      25-Jul-1984
0000 51         Add optional bugcheck on message free queue empty.
0000 52
0000 53         V03-025 NPK3055      N. Kronenberg      14-Jul-1984
0000 54         Fix bug in RSP_UNREC_PKT that was wiping out R1
0000 55         prior to logging unrecognized pkt.
0000 56
0000 57         V03-024 NPK3053      N. Kronenberg      24-May-1984

```



```

0000 58 :
0000 59 :
0000 60 :
0000 61 :
0000 62 :
0000 63 :
0000 64 :
0000 65 :
0000 66 :
0000 67 :
0000 68 :
0000 69 :
0000 70 :
0000 71 :
0000 72 :
0000 73 :
0000 74 :
0000 75 :
0000 76 :
0000 77 :
0000 78 :
0000 79 :
0000 80 :
0000 81 :
0000 82 :
0000 83 :
0000 84 :
0000 85 :
0000 86 :
0000 87 :
0000 88 :
0000 89 :
0000 90 :
0000 91 :
0000 92 :
0000 93 :
0000 94 :
0000 95 :
0000 96 :
0000 97 :
0000 98 :
0000 99 :
0000 100 :
0000 101 :
0000 102 :
0000 103 :
0000 104 :
0000 105 :
0000 106 :
0000 107 :
0000 108 :
0000 109 :
0000 110 :
0000 111 :
0000 112 :
0000 113 :
0000 114 :

```

Fix problem with clearing response bit in a REQID immediately requeued to poll single remaining good path.

V03-023 NPK3048 N. Kronenberg 9-Apr-1984
Fix bug in calculation of path select in reissuing of REQID in RSP_PATH_FAIL.
Change aux status from SSS_POWERFAIL (which was a fib anyway) to SSS_VCCLOSED when SETCKT closed is done.
Support two new response status subtypes for response sequence number mismatch and for sequenced msg received on closed VCD. Both status types are presently implemented in experimental ucode only.

V03-022 NPK3047 N. Kronenberg 15-Mar-1984
Modify RSP_PATH_FAIL in the case where only one path is discovered bad by a REQID. For this case, immediately reuse the REQID response to send a second REQID over the other path to determine if it is also bad. This is an optimization in discovering bad vc's via polling.

V03-021 NPK3046 N. Kronenberg 7-Mar-1984
Augment the comments in the INT\$... port operation primitives at the start of this module.
Restore the conditional calls to TRC\$LOGMSG for sent datagrams and messages.
Zero R2 following the queuing of a command to the port to show that the software no longer owns the packet (helps catch bugs.)
Change the R2 input to INT\$READCNT to be the start of the PPD layer of the pkt rather than the start of the 'application' data which is really not applicable.
Fix receipt of an unrecognized packet to do nothing if the opcode of the packet is either maintenance reset or start since these arrive in the normal course of events.
Remove extra instructions in INT\$SNDMSG/SNDGMSL which were computing the response bit rather than using it directly.

V03-020 TMK0001 Todd M. Katz 21-Feb-1984
Make the following changes to PA\$INT, the PADRIVER's interrupt service routine:

1. Fix a bug in the processing of errors reported via PSR defined interrupts. What should happen when a defined PSR interrupt occurs is that it should be logged as a non-specific error, and the port should be re-initialized. Instead, because of a BEQL which should have been a BNEQ, such errors were being treated as un-defined errors (and correspondingly un-defined errors were being treated as defined errors). Un-defined errors result in an optional bugcheck followed by the logging of an un-expected interrupt and an attempt to re-init the port.
2. Port initialization has been changed so that it is now done


```

0000 115 :
0000 116 :
0000 117 :
0000 118 :
0000 119 :
0000 120 :
0000 121 :
0000 122 :
0000 123 :
0000 124 :
0000 125 :
0000 126 :
0000 127 :
0000 128 :
0000 129 :
0000 130 :
0000 131 :
0000 132 :
0000 133 :
0000 134 :
0000 135 :
0000 136 :
0000 137 :
0000 138 :
0000 139 :
0000 140 :
0000 141 :
0000 142 :
0000 143 :
0000 144 :
0000 145 :
0000 146 :
0000 147 :
0000 148 :
0000 149 :
0000 150 :
0000 151 :
0000 152 :
0000 153 :
0000 154 :
0000 155 :
0000 156 :
0000 157 :
0000 158 :
0000 159 :
0000 160 :
0000 161 :
0000 162 :
0000 163 :
0000 164 :
0000 165 :
0000 166 :
0000 167 :
0000 168 :
0000 169 :
0000 170 :
0000 171 :

```

at fork IPL instead of at IPL\$POWER. Therefore, on a power-up when it comes time to re-init the port, first device interrupts are disabled by clearing the hardware state and placing the port in the un-initialized state. Next, the new routine INIS\$FORK is called with the address of INIS\$PORT, the port initialization routine which is to eventually be executed at fork IPL. INIS\$FORK will create a fork process and then returns control to PAS\$INT so that the interrupt can be dismissed. In the creation of this fork process, INIS\$FORK knows how to extract the fork block from the appropriate fork queue in an atomic fashion, and how to make proper use of the fork block interlock bit. When the fork process resumes, it does so within INIS\$FORK, and it proceeds to jump to INIS\$PORT so that a re-initialization of the port can be done.

3. At the present time if the interrupt service routine has decided to crash and then re-initialize the port it will go ahead and do so regardless of whether or not the port is already in the process of being crashed and re-initialized. This is incorrect. The port should only be crashed and re-initialized once. If the port driver is already in the process of informing the SYSAPs as a prelude to crashing and re-initializing the port, it should not do so a second time. Therefore, if PDT\$V_PWF_CLNUP is set within PDT\$V_LPORT_STS, indicating that the port driver is in the process of informing the SYSAPs and will crash and re-initialize the port after the last connection is DISCONNECTed, do not instigate a second round of SYSAP notification, port crashing, and port re-initialization.

V03-019 SRB0113 Steve Beckhardt 20-Feb-1984
Modified INT\$ALLOC_MSG so that this CDRP waits for pool if any CDRPS are already waiting for pool. Modified INT\$DEAL_MSG, INT\$DEAL_DG, etc. to call EXE\$DEANONPAGED directly rather than going through COM\$DRVDEALMEM.

V03-018 NPK3045 N. Kronenberg 23-Feb-1984
Remove instruction in RSP_CLOSED_VC which set PB\$C_VC_FAIL prior to calling SC\$VCCLOSED. Must let SC\$VCCLOSED do it instead.

V03-017 NPK3042 N. Kronenberg 6-Feb-1984
Change INT\$CLRCACHE to simply return if the CLSCKT_DG is not available.
Change REC_SETCKT to call SC\$SETCKT_CLSD instead of SC\$VCCLOSED.

V03-016 NPK3039 N. Kronenberg 11-Jan-1984
Add comments in INT\$CLRCACHE.
When cache clear is sent, zero PB\$CLSCKT_DG to show the packet belongs to port now. When cache clear or SETCKT comes back, restore pkt address to PB\$CLSCKT_DG. Fix PSR error check to distinguish between a defined bit is set and an undefined bit is set (difference in handling is in error log type.) Add \$DEBUGCHECK if undefined bit is set. (This bugcheck is being used with a test version of ucode to report unrecognized


```

0000 172 :
0000 173 :
0000 174 :
0000 175 :
0000 176 :
0000 177 :
0000 178 :
0000 179 :
0000 180 :
0000 181 :
0000 182 :
0000 183 :
0000 184 :
0000 185 :
0000 186 :
0000 187 :
0000 188 :
0000 189 :
0000 190 :
0000 191 :
0000 192 :
0000 193 :
0000 194 :
0000 195 :
0000 196 :
0000 197 :
0000 198 :
0000 199 :
0000 200 :
0000 201 :
0000 202 :
0000 203 :
0000 204 :
0000 205 :
0000 206 :
0000 207 :
0000 208 :
0000 209 :
0000 210 :
0000 211 :
0000 212 :
0000 213 :
0000 214 :
0000 215 :
0000 216 :
0000 217 :
0000 218 :
0000 219 :
0000 220 :
0000 221 :
0000 222 :
0000 223 :
0000 224 :
0000 225 :
0000 226 :
0000 227 :
0000 228 :

```

command via interrupt instead of later via response status.)
Fix INV_OPCODE or unexpected opcode handling to
look up path block before branching to RSP_CRASH_PORT.
On pkt size violation with received seq msg type opcode
(MSGREC, RETDAT, RETCNF) port does not seem to close
the vc, so close it explicitly.

- V03-015 NPK3037 N. Kronenberg 11-Nov-1983
Optimize performance in interrupt service.
Add \$DEBUGCHECKS for various packet status errors
which normally result in crashing and reiniting
the port.
- V03-014 NPK3029 N. Kronenberg 18-Jul-1983
Enhancements for V4.0:
Change action on buffer memory system error (e.g.,
port detected RDS) to handle port vc closure instead
of crashing the port.
- V03-013 NPK3028 N. Kronenberg 19-Jul-1983
Fix INT\$TRNMSG to turn scs response around on high
priority queue.
- V03-012 NPK3024 N. Kronenberg 18-May-1983
Add variable network header logic.
- V03-011 KTA3046 Kerbey T. Altmann 28-Mar-1983
Redo for SCS/PPD split.
- V03-010 NPK3012 N. Kronenberg 29-Nov-1982
Fix cache clear response error to BSBW to ERR\$CACHECLR
instead of JMP.
- V03-009 NPK3010 N. Kronenberg 11-Nov-1982
Invoke \$SYSAPDEF and use symbols for dg rcv and
sent flags.
- V03-008 NPK3004 N. Kronenberg 30-Jul-1982
Add comments for interrupt service. Add service for
CI750 specific errors.
- V03-007 NPK3002 N. Kronenberg 1-Jul-1982
Fix interrupt service to dismiss PUP interrupt in the
case where the PDT has been removed.
- V03-006 ROW0113 Ralph O. Weber 28-JUN-1982
Change PASINT to log error with a hardware error subtype of
"unexpected interrupt" when either PIC or MFQE is set in
PA_PS.
This change will be in a new driver image shipped in V3.1.
- V03-005 NPK3001 N. Kronenberg 22-Jun-1982
Fixed cleanup of PPD type SNDDG with error status to
return to pool or dg free queue depending on response
bit.
- V03-004 ROW0095 Ralph O. Weber 7-JUN-1982

0000 229 :
0000 230 :
0000 231 :
0000 232 :
0000 233 :
0000 234 :
0000 235 :
0000 236 :
0000 237 :
0000 238 :
0000 239 :
0000 240 :
0000 241 :
0000 242 :
0000 243 :
0000 244 :
0000 245 :
0000 246 :
0000 247 :
0000 248 :
0000 249 :
0000 250 :
0000 251 :
0000 252 :
0000 253 :
0000 254 :
0000 255 :
0000 256 :
0000 257 :
0000 258 :
0000 259 :
0000 260 :
0000 261 :--

Add calls to error logging routines in the interrupt service routine at REINIT_PORT, PWR_DN, and PWR_UP as well as to the following routines: RSP_PATH_FAIL, RSP_UNREC_PKT, RSP_CLOSED_VC, RSP_CRASH_PORT, RSP_CRASH_VC, and all the INISFATALQ error entry points. Also added necessary reference to SPAERDEF and rearranged the queue error entry point code to make it simpler.
This change will be in a new driver image shipped in V3.1.

V03-003 NPK2019 N. Kronenberg 6-Apr-1982

Add routine RSP_CRASH_PORT.
Fixed a number of error conditions in responses which previously bugchecked to call RSP_CRASH_PORT.
Added separate error handlers for queue interlock failures on different queues. Change failure to crash port and continue rather than bugcheck.
Removed test code from REM_NEXT_RSP.
Change RSP_UNREC_PKT to crash vc if unrecognized pkt is from node we have open VC to.
Fix RSP action dispatcher to branch to error handler and not to leave anything on the stack.

V03-002 NPK2018 N. Kronenberg 25-Mar-1982

Fix calculation of path status byte in RSP_PATH_FAIL.
Fix REC_SETCKT to not deallocate dg if this is a SETCKT closed on VC crash. This dg is attached to PB.
Fix to permit path failures reported on REQID, SNDRST, SNDSTRT to cause virtual circuit crashes.

V03-001 NPK2016 N. Kronenberg 18-Mar-1982

Fixed .TITLE

DEFINITIONS

```

0000 263      .SBTTL  DEFINITIONS
0000 264
0000 265  ::
0000 266  :: Set PSECT to driver code:
0000 267  ::
0000 268
00000000 269      .PSECT  $$$115_DRIVER, LONG
0000 270
0000 271  ::
0000 272  :: System definitions (LIB.MLB):
0000 273  ::
0000 274
0000 275      .nocross
0000 276      $DYNDEF          ; Structure type definitions
0000 277      $IDBDEF         ; Interrupt Dispatch Block format
0000 278      $NDTDEF         ; Nexus Device Type codes
0000 279      $PBDEF          ; Path Block format
0000 280      $PDTDEF         ; Port Descriptor Table format
0000 281      $SSDEF          ; System service definitions
0000 282      $SYSAPDEF       ; Send/recv dg flags
0000 283      $UCBDEF         ; Unit Control Block format
0000 284
0000 285  ::
0000 286  :: CI specific definitions (PALIB.MLB):
0000 287  ::
0000 288
0000 289      $PAERDEF          ; Port driver error code values
0000 290      $PAPBDEF         ; CI extension to PB
0000 291      $PAPDTDEF        ; CI extension to PDT
0000 292      $PAREGDEF         ; Define ci register layout
0000 293      $PPDDEF          ; Port-port message format
0000 294      $PAUCBDEF        ; UCB extensions
0000 295      .cross

```

PPD SEND ROUTINES

```
0000 297 .SBTTL PPD SEND ROUTINES
0000 298 .SBTTL - SNDDG SEND DATAGRAM
0000 299
0000 300 :+
0000 301 : These routines are used to send out a datagram buffer. The entry
0000 302 : points are different for the position of R2 on entry - SNDDG has
0000 303 : R2 pointing to SYSAP portion of message, SNDDG1 has R2 pointing
0000 304 : to the PPD/SCS portion.
0000 305
0000 306 Inputs:
0000 307
0000 308 R0 -Input flag
0000 309 R1 -Addr of PB
0000 310 R2 -Addr of buffer (see comments above)
0000 311 R4 -Addr of PDT
0000 312
0000 313 Outputs:
0000 314
0000 315 R0 -Destroyed
0000 316 R2 -Zero to show pkt now owned by port
0000 317
0000 318 Other registers -Preserved
0000 319
0000 320 :-
0000 321
0000 322 :
0000 323 : Table of correct settings of RETFLAG and DISPOSAL flag respectively
0000 324 : for each option the SYSAP can specify in R0:
0000 325 :
0000 326
0000 327 ASSUME SYSAP$C_DISPQ EQ 0
0000 328 ASSUME SYSAP$C_DISPRET EQ 1
0000 329 ASSUME SYSAP$C_DISPPQ EQ 2
0000 330
0000 331 DG_SENT_FLGS:
0000 332
0000 333 .BYTE 0,0 ; R0 = SYSAP$C_DISPQ, return sent
0000 334 ; dg to port free queue
0000 335 .BYTE PPD$M_RSP,PPD$M_DISPOSE ; R0 = SYSAP$C_DISPRET, return sent
0000 336 ; dg to response queue, then to sysap
0000 337 .BYTE PPD$M_RSP,0 ; R0 = SYSAP$C_DISPPQ, return sent
0000 338 ; dg to response queue, then to pool
0000 339
0000 340
0000 341 INT$SNDDG::
0000 342
0000 343 52 0190 C4 C2 0006 343 SUBL2 PDT$L_DGHDRSZ(R4),R2 ; Point to PPD start of buffer
0000 344 03 B0 000B 344 MOVW #PPD$C_SCS_DG,- ; Set PPD type to application
0000 345 12 A2 000D 345 PPD$W_MTYPE(R2) ; datagram
0000 346
0000 347 INT$SNDDG1::
0000 348
0000 349 50 ED AF40 3E 000F 349 MOVAV DG_SENT_FLGS[R0],R0 ; Get address of RETFLAG/DISPOSE
0000 350 00010000 8F C9 0014 350 BISL3 #<PPD$C-SNDDG@16>,- ; Set opcode,
0000 351 OC A1 001A 351 PPSB_RSTATION(R1),- ; and port,
0000 352 OC A2 001C 352 PPD$B_PORT(R2) ; into header
0000 353 OF A2 80 88 001E 353 BISB (R0)+,PPD$B_FLAGS(R2) ; Set RETFLAG
```


- SNDDG SEND DATAGRAM

OB A2	60	90	0022	354	MOVB	(R0),PPD\$B_SWFLAG(R2)	; Set DISPOSE
			0026	355			
			0026	356	.IF	DF PASDEBUG	; If debug is enabled,
			0026	357	BSBW	TRC\$LOGMSG	; log command in trace buffer
			0026	358	.ENDC		
			0026	359			
17	11		0026	360	BRB	QHI	; Send it

- SNDMSG SEND SEQUENCED MESSAGE

```
0028 362 .SBTTL - SNDMSG SEND SEQUENCED MESSAGE
0028 363
0028 364 ;+
0028 365 : INT$SNDMSG formats the PPD header and sends a sequenced message at
0028 366 : high priority. INT$SNDMSG is the same, but sends the message at
0028 367 : low priority. In both cases the input flag specified in R0 has two
0028 368 : legal values, 0 and 1. If 0, the port is instructed to return the sent
0028 369 : message to the port free queue. If 1, the port is told to return
0028 370 : the message to the response queue for disposal by SCS.
0028 371
0028 372 : Inputs:
0028 373
0028 374 : R0 -Input flag
0028 375 : R1 -Addr of PB
0028 376 : R2 -Addr of buffer
0028 377 : R4 -Addr of PDT
0028 378
0028 379 : Outputs:
0028 380
0028 381 : R0 -Destroyed
0028 382 : R2 -Zero to show pkt now owned by port
0028 383
0028 384 : Other registers -Preserved
0028 385
0028 386 :-
0028 387
0028 388 .ENABLE LSB
0028 389
0028 390 INT$SNDMSG::
0028 391
52 00B4 C4 C2 0028 392 SUBL2 PDT$ MSGHDRSZ(R4),R2 ; Point to PPD start of buffer
0028 393 MOVW #PPD$ SCS MSG,- ; Set PPD type to application
12 A2 B0 002D 394 PPD$W MTYPE(R2) ; message
00020000 8F C9 002F 395 BISL3 #<PPD$C SNDMSG@16>,- ; Set opcode,
0C A1 0037 396 PPSB RSTATION(R1),- ; and port,
0C A2 0039 397 PPD$B PORT(R2) ; into header
OF A2 50 88 003B 398 BISB R0,PPD$B_FLAGS(R2) ; Set RETFLAG
003F 399
003F 400 .IF DF PASDEBUG ; If debug enabled,
003F 401 BSBW TRC$LOGMSG ; log message in trace buffer
003F 402 .ENDC
003F 403
003F 404 INT$INS_COMQH::
003F 405
003F 406 QHI: $INS_COMQH HIGH ; Send it out
52 D4 005A 407 CLRL R2 ; Zero buffer pointer
005C 408 RSB ; Return
05 005D 409
005D 410 INT$SNDMSG::
005D 411
52 00B4 C4 C2 005D 412 SUBL2 PDT$ MSGHDRSZ(R4),R2 ; Point to PPD start of buffer
0028 393 MOVW #PPD$ SCS MSG,- ; Set PPD type to application
12 A2 B0 0062 413 PPD$W MTYPE(R2) ; message
00020000 8F C9 0064 414 BISL3 #<PPD$C SNDMSG@16>,- ; Set opcode,
0C A1 0066 415 PPSB RSTATION(R1),- ; and port,
0C A2 006C 416 PPD$B PORT(R2) ; into header
OF A2 50 88 006E 417 BISB R0,PPD$B_FLAGS(R2) ; Set RETFLAG
0070 418
```


- SNDMSG SEND SEQUENCED MESSAGE

```

0074 419
0074 420      .IF      DF PA$DEBUG      ; If debug enabled,
0074 421      BSBW      TRC$LOGMSG      ; log message in trace buffer
0074 422      .ENDC      ;
31 11 0074 423      BRB      QLOW      ; Send it out
0076 424
0076 425      INT$TRNMSG::
0076 426
0076 427      SUBL2      PDT$L MSGHDRSZ(R4),R2 ; Point to PPD start of buffer
0076 428      INSV      #<PPD$C_SNDMSG@8>,#0,- ; Set opcode
007B 429      #24,PPD$B_STATUS(R2) ; Into header
0082 430
0085 431      .IF      DF PA$DEBUG      ; If debug enabled,
0085 432      BSBW      TRC$LOGMSG      ; log message in trace buffer
0085 433      .ENDC      ;
0085 434
B8 11 0085 435      BRB      QHI      ; Send it out
0087 436
0087 437      .DSABL      LSB
0087 438

```


- REQDAT REQUEST BLOCK DATA

```

0087 440 .SBTTL - REQDAT REQUEST BLOCK DATA
0087 441
0087 442 :+
0087 443 : INT$REQDAT formats the PPD header of a message for a request block
0087 444 : transfer operation and queues the message to the port.
0087 445 :
0087 446 : Inputs:
0087 447 :
0087 448 : R1 -Addr of PB
0087 449 : R2 -Addr of message buffer
0087 450 : R4 -Addr of PDT
0087 451 :
0087 452 : Outputs:
0087 453 :
0087 454 : R0 -Destroyed
0087 455 : R2 -Zero to show pkt now owned by port
0087 456 :
0087 457 : Other registers -Perserved
0087 458 :
0087 459 :-
0087 460
0087 461 INT$REQDAT::
0087 462
52 00B4 C4 C2 0087 463 SUBL2 PDT$ MSGHDRSZ(R4),R2 ; Point to PPD start of buffer
00080000 8F C9 008C 464 BISL3 #PPD$ REQDATA@16,- ; Set opcode
OC A1 0092 465 PBSB R$STATION(R1),- ; and dst port
OC A2 0094 466 PPD$B _PORT(R2) ; in msg header
OF 11 0096 467 BRB QLOW ; Join common code

```

- SNDDAT SEND BLOCK DATA

```

0098 469 .SBTTL - SNDDAT SEND BLOCK DATA
0098 470
0098 471 ;+
0098 472 ; INT$SNDDAT formats the PPD header of a message for a send block
0098 473 ; transfer operation and queues the message to the port.
0098 474 ;
0098 475 ; Inputs:
0098 476 ;
0098 477 ; R1 -Addr of PB
0098 478 ; R2 -Addr of message buffer
0098 479 ; R4 -Addr of PDT
0098 480 ;
0098 481 ; -
0098 482 ;
0098 483 INT$SNDDAT::
0098 484
52 00B4 C4 C2 0098 485 SUBL2 PDT$MSGHDRSZ(R4),R2 ; Point to PPD start of buffer
00100000 8F C9 009D 486 BISL3 #PPD$SNDDAT@16,- ; Set opcode
OC A1 00A3 487 PB$RSTATION(R1),- ; and dest port #
OC A2 00A5 488 PPD$PORT(R2) ; in msg header
00A7 489
00A7 490 INT$INS_COMQL::
00A7 491
52 D4 00A7 492 QLOW: $INS_COMQLOW ; Send off xfer command
05 00C2 493 CLRL R2 ; Zero buffer pointer
00C4 494 RSB ; Return

```


- MRESET MAINTENANCE RESET

```

00EC 539      .SBTTL -      MRESET MAINTENANCE RESET
00EC 540
00EC 541      :+
00EC 542      : INTSMRESET formats the PPD header of a datagram to do a remote
00EC 543      : port maintenance reset and queues the dg to the local port.
00EC 544
00EC 545      : Inputs:
00EC 546
00EC 547      :      R0      -0/1 for don't/do force reset
00EC 548      :      R1      -Addr of remote station to reset
00EC 549      :      R2      -Addr of datagram
00EC 550      :      R4      -Addr of PDT
00EC 551
00EC 552      : Outputs:
00EC 553
00EC 554      :      R0      -Status: SSS_NORMAL, SSS_NOSUCHNODE
00EC 555      :      R1      -Destroyed
00EC 556      :      R2      -0, to show pkt now owned by port
00EC 557
00EC 558      : Other registers      -Preserved
00EC 559
00EC 560      :-
00EC 561
00EC 562      INTSMRESET::
00EC 563
00EC 564      MOVZBL (R1),R1      ; Get remote port #
00EF 565      CMPB R1,PDT$B_MAX_PORT(R4)      ; Legal?
00F4 566      BGTRU PORT_ERR      ; No, error out
00F6 567      ROTL #-1,R0,R0      ; Reposition force bit in h.o.
00FB 568      ; bit of flags and
00FB 569      BISL R0,R1      ; OR into port #
00FE 570      SUBL2 PDT$B_DGHDRSZ(R4),R2      ; Point to PPD start of buffer
0103 571      BISL3 #<<PPD$M_RSP@24>>!,      ; Specify response wanted, and
0104 572      <PPD$C_SNDRST@16>>,-      ; OR in opcode
0104 573      R1,PPD$B_PORT(R2)      ; and put in dg
010C 574      CLRQ PPD$Q_XCT_ID(R2)      ; Set transaction ID = 0
010F 575      BRB DO_Q      ; Send it

```


- MSTART SEND MAINTENANCE START

.SBTTL - MSTART SEND MAINTENANCE START

```

0111 577      INT$MSTART formats the PPD header of a datagram to do a remote port
0111 578      maintenance start and queues the dg to the local port.
0111 579      :+
0111 580      :
0111 581      : Inputs:
0111 582      :
0111 583      :   R0      -1/0 for use default/specified start addr
0111 584      :   R1      -Addr of remote station
0111 585      :   R2      -Addr of datagram
0111 586      :   R3      -Start address to send
0111 587      :   R4      -Addr of PDT
0111 588      :
0111 589      : Outputs:
0111 590      :
0111 591      :   R0      -Status: SS$_NORMAL, SS$_NOSUCHNODE
0111 592      :   R1      -Destroyed
0111 593      :   R2      -0, to show pkt now owned by port
0111 594      :
0111 595      :   Other registers      -Preserved
0111 596      :
0111 597      :
0111 598      :
0111 599      :-
0111 600
0111 601 INT$MSTART::
0111 602
0111 603      MOVZBL (R1),R1      ; Get remote port #
0111 604      CMPB  R1,PPD$B_MAX_PORT(R4) ; Legal?
0111 605      BGTRU  PORT_ERR      ; No, error out
0111 606      ROTL  #-1,R0,R0      ; Position default start addr flag
0111 607      BISL  R0,R1      ; and OR into port #
0111 608      SUBL2  PPD$B_DGHDRSZ(R4),R2 ; Point to PPD start of buffer
0111 609      BISL3  #<<PPD$M_RSP@24>>,- ; Specify response wanted, and
0111 610      <PPD$C_SNDSTRT@16>>,- ; OR in opcode
0111 611      R1,PPD$B_PORT(R2) ; and put in dg
0111 612      MOVL  R3,PPD$B_ST_ADDR(R2) ; Set start addr in case used
0111 613      CLRQ  PPD$Q_XCT_ID(R2) ; Set transaction ID = 0
0111 614      BRB  DO_Q      ; Send it

```

```

          51  61  9A 0111 603
017C C4 51 91 0114 604
          CB 1A 0119 605
50 50 FF 8F 9C 011B 606
          51 50 C8 0120 607
52 0190 C4 C2 0123 608
          C9 0128 609
          0129 610
OC A2 51 01070000 8F 0129 611
          18 A2 53 D0 0131 612
          10 A2 7C 0135 613
          A6 11 0138 614

```

```

013A 616      .SBTTL -      CLRCACHE,      CLEAR ANY PPD LAYER CACHES
013A 617
013A 618      :+
013A 619      : Sends out a sequenced msg at the lowest priority asking for a
013A 620      : response. It is assumed that the msg completes with circuit closed
013A 621      : error status. The msg is sent after any other traffic on this
013A 622      : circuit. When the msg comes back, it must be after any other
013A 623      : commands queued to the port or responses held by the port and hence
013A 624      : guarantees that the port holds no more packets associated with this
013A 625      : circuit.
013A 626
013A 627      : The packet used to send the cache clear should always be available
013A 628      : since this routine is not supposed to be called if a virtual circuit
013A 629      : failure is already in progress. However, just in case, if the packet
013A 630      : is not available simply return.
013A 631
013A 632      : Inputs:
013A 633
013A 634      :      R1      -PB addr
013A 635
013A 636      : Outputs:
013A 637
013A 638      :      R0,R2      -Destroyed
013A 639      :      Other registers      -Preserved
013A 640
013A 641      :      PB$CL_SCKT_DG(R1)      -0, to show port owns packet
013A 642      :-
013A 643
013A 644      .ENABL  LSB
013A 645
013A 646      INT$CLRCACHE::
013A 647
013A 648      MOVL  PB$CL_SCKT_DG(R1),R2      : Get addr of preallocated dg
013A 649      BEQL  10$                      : Branch if not available
013A 650      CLRL  PB$CL_SCKT_DG(R1)        : Zero address of dg in PB
013A 651      MOVW  #PPD$C_CACHE_LEN,-      : Set PPD length
013A 652      PPDSW LENGTH(R2)
013A 653      MOVW  #PPD$C_CACHECLR,-      : Set PPD type code
013A 654      PPDSW MTYPE(R2)
013A 655      MOVB  PB$B_RSTATION(R1),-      : Get remote port number from PB
013A 656      PPDSW PORT(R2)
013A 657      INSV  #<PPD$M_RSP@16>!-      : Send msg, specifying response
013A 658      <PPD$C_SNDMSG@8>,-
013A 659      #0,#24,PPD$B_STATUS(R2)      : to offending port
013A 660      BRW  QLOW                      : at low priority so it
013A 661
013A 662
013A 663
013A 664
013A 665
013A 666
013A 667
013A 668
013A 669      10$:  RSB                      : Return
013A 670
013A 671      .DSABL  LSB

```

52 54 A1 D0 013A 648 MOVL PB\$CL_SCKT_DG(R1),R2 : Get addr of preallocated dg
1F 13 013E 649 BEQL 10\$: Branch if not available
54 A1 D4 0140 650 CLRL PB\$CL_SCKT_DG(R1) : Zero address of dg in PB
02 B0 0143 651 MOVW #PPD\$C_CACHE_LEN,- : Set PPD length
10 A2 0145 652 PPDSW LENGTH(R2)
8000 8F B0 0147 653 MOVW #PPD\$C_CACHECLR,- : Set PPD type code
12 A2 014B 654 PPDSW MTYPE(R2)
0C A1 90 014D 655 MOVB PB\$B_RSTATION(R1),- : Get remote port number from PB
0C A2 0150 656 PPDSW PORT(R2)
F0 0152 657 INSV #<PPD\$M_RSP@16>!- : Send msg, specifying response
0D A2 18 00 00010200 8F 0153 658 <PPD\$C_SNDMSG@8>,-
FF48 31 015C 659 #0,#24,PPD\$B_STATUS(R2) : to offending port
015F 660 BRW QLOW : at low priority so it
015F 661
015F 662
015F 663
015F 664
015F 665
015F 666
015F 667
015F 668
05 015F 669 10\$: RSB : Return
0160 670
0160 671 .DSABL LSB

PA_INTERRUPT_SERVICE ROUTINE

```

0160 673      .SBTTL PA_INTERRUPT_SERVICE ROUTINE
0160 674
0160 675 :+
0160 676 : PASINT is called to service a CI interrupt. First, the shortest checks
0160 677 : possible are made to determine that the interrupt is for a response,
0160 678 : not an error. Three checks are necessary. If:
0160 679 :
0160 680 : - there are no bits set in the configuration register (CNF) besides
0160 681 :   the adapter type code, and
0160 682 : - MTE is clear in the port status register (PS), and
0160 683 : - there are no bits except response available (RQA) set in the PS, then
0160 684 :
0160 685 : no error has occurred and fork is taken to dequeue the next response.
0160 686 :
0160 687 : If there are bits set in the CNF besides the adapter type code, then
0160 688 : analyze and handle those bits as follows:
0160 689 :
0160 690 : 1. If any 11/780 SBI error bits (31:26) or C are set, then write
0160 691 :   the CNF to itself to clear the error condition and proceed to
0160 692 :   the MTE check. The rationale for ignoring CRDs is that memory is
0160 693 :   still considered good in the presence of corrected read errors. The
0160 694 :   rationale for ignoring SBI errors is that they will be seen by
0160 695 :   appropriate SBI interrupts.
0160 696 :
0160 697 : 2. If any of the bits CXTMO, RDTO, CXTER, RDS, TFAIL, TDEAD, PFD, or
0160 698 :   the 11/750 specific bits CTO, CIBPE, or MAINT are set in the CNF, then
0160 699 :   reinit the port. The assumption is that the port has a serious error.
0160 700 :   The assumption may be too optimistic as the CPU may be at fault, but
0160 701 :   this handling results in minimum disturbance to running software if
0160 702 :   it is true. Note that CXTMO on the 780 corresponds to NXM on the 750;
0160 703 :   RDS on the 780 corresponds to UCE on the 750; and TDEAD/TFAIL on the
0160 704 :   780 correspond to T DCLO/T ACLO on the 750.
0160 705 :
0160 706 : 3. If PDN is set in the CNF, log it, write CNF to itself to clear the
0160 707 :   condition, and initiate software power fail recovery logic.
0160 708 :
0160 709 : 4. If PUP is set in the CNF, log it, disable CI interrupts, and execute
0160 710 :   port reinitialization.
0160 711 :
0160 712 : 5. If the C1750 bit, NOCI, is set, then log registers and shut the port
0160 713 :   down without bothering to do any retries.
0160 714 :
0160 715 : 6. If any other bits (undefined) are set, write the CNF to itself to
0160 716 :   clear the condition and proceed to the MTE check.
0160 717 :
0160 718 : Next check MTE in the PS. If it is set, then some sort of parity error is
0160 719 : implied. In this case, device registers are logged, and the port is
0160 720 : reinitialized.
0160 721 :
0160 722 : Finally, if MTE is clear, then the rest of the PS can be analyzed. If
0160 723 : there are any bits set in the PS besides RQA, then the device registers
0160 724 : are logged and the port reinitialized.
0160 725 :
0160 726 :
0160 727 : Call: JSB from CRB interrupt vector dispatcher
0160 728 : Inputs:
0160 729 :

```


PA_INTERRUPT_SERVICE ROUTINE

```
0160 730 :      @0(SP)          -Addr of IDB
0160 731 :      4(SP)-16(SP)    -Saved R2-R5
0160 732 :      20(SP)         -Interrupt PC
0160 733 :      24(SP)         -Interrupt PSL
0160 734 :
0160 735 :-
0160 736
0160 737 ASSUME PA_CNF EQ 0
0160 738 ASSUME PA_PS_M_MTE EQ 1@31
0160 739
FC000000 0160 740 SBIERR  =^XFC000000      ; CNF, SBI errors that are ignored
0160 741      ; because they will be caught via
0160 742      ; SBI fault: PAR FLT, WSQ FLT,
0160 743      ; URD FLT, unused bit, MXT FLT,
0160 744      ; XMT FLT
0160 745
001EE700 0160 746 FATAL_CNFERR =^X001EE700      ; CNF errors that are not ignored:
0160 747      ; CXTMO, RDTO, CXTER, RDS, TFAIL,
0160 748      ; TDEAD, PFD, and the 750 specific
0160 749      ; errors MAINT, CIBPE and CTO
0160 750
0000007E 0160 751 PSR_ERRORS =^X0000007E      ; Defined bits in the Port Status
0160 752      ; register.
0160 753
0160 754      .ENABL  LSB
0160 755
0160 756 PASINT::
0160 757
53 9E D0 0160 758      MOVL      @0(SP)+,R3      ; Get IDB address
54 63 D0 0163 759      MOVL      IDBSL_CSR(R3),R4      ; Get addr of configuration register
55 18 A3 D0 0166 760      MOVL      IDBSL_UCBLST(R3),R5      ; Get UCB addr
016A 761
016A 762 CHK_CNF:
016A 763
38 64 D1 016A 764      CMPL      PA_CNF(R4),#NDTS_CI      ; Any config reg bits set except type?
27 12 016D 765      BNEQ      CNF_ERR      ; Branch if yes
016F 766
016F 767 CHK_MTE:
016F 768
52 0900 C4 D0 016F 769      MOVL      PA_PS(R4),R2      ; MTE set? (Get PSR in general register)
03 14 0174 770      BGTR      CHR_PSR      ; Branch if not
00A7 31 0176 771      BRW      REINIT_PORT      ; Else parity error forces reinit
0179 772
0179 773 CHK_PSR:
0179 774
52 FFFFFFFE 8F D3 0179 775      BITL      #^C<PA_PS_M_RQA>,R2      ; Any bits set in PSR besides RQA?
25 12 0180 776      BNEQ      PS_ERR      ; Branch if so (serious error)
0182 777      ; Else no error!
0182 778      MOVL      #PA_PSR_M_PSC,-      ; Release port registers
0918 C4 D0 0184 779      PA_PSR(R4)
01 01 E2 0187 780      BBSS      #UCB_V_FKLOCK,-      ; Set fork block interlock and
68 A5 0189 781      UCB$Q_DEVSTS(R5),-      ; branch if already set
03 018B 782      DISMISS_INT      ; to dismiss interrupt
010D 30 018C 783      BSBW      HANDLE_INT      ; Handle interrupt at fork IPL
018F 784
018F 785 DISMISS_INT:
018F 786
```


PA_INTERRUPT_SERVICE ROUTINE

```

      52 8E 7D 018F 787      MOVQ      (SP)+,R2      ; Restore registers saved
      54 8E 7D 0192 788      MOVQ      (SP)+,R4      ; on normal interrupt
                    02 0195 789      REI
                    0196 790
                    0196 791      CNF_ERR:
                    0196 792
                    0196 793
52 52 64 D0 0196 794      MOVL      PA_CNF(R4),R2      ; Get copy of config register handy
FC010000 8F D3 0199 795      BITL      #<SBIERR!PA_CNF_M_CRD>,R2 ; Ignorable error bit set?
      5C 13 01A0 796      BEQL      OTHER_CNF_ERR    ; Branch if not
                    01A2 797
                    01A2 798      CNF_OK:              ; Else config register OK
                    01A2 799
      64 64 D0 01A2 800      MOVL      PA_CNF(R4),PA_CNF(R4) ; Clear status bits in CNF
      C8 11 01A5 801      BRB      CHR_MTE           ; Continue error checking
                    01A7 802
                    01A7 803      PS_ERR:
                    01A7 804
25 52 01 E1 01A7 805      BBC      #PA_PS_V_MFQE,R2,5$    ; Branch if not MFQ empty
      7E 50 7D 01AB 806      MOVQ      R0,-(SP)         ; Else save more registers
54 0084 C5 D0 01AE 807      MOVL      UCB$$_PDT(R5),R4    ; Get PDT addr
      54 63 D0 01B3 808      $DEBUGCHECK #ERR$V_DEB_MFQE  ; Do optional bugcheck
      01C6 809      MOVL      IDB$$_CSR(R3),R4          ; Retrieve config register
      01C9 810      ; addr if bugcheck disabled
50 8002 8F 32 01C9 811      CVTWL     #<PAERSK_ES_HWER ! ^X8000>,R0 ; Set up and log
      58 11 01CE 812      BRB      HDWR_ERR_CODE        ; general hw error
      01D0 813
52 0000007E 8F D3 01D0 814 5$:      BITL      #PSR_ERRORS,R2 ; Is error in PSR an
      01D7 815      ; defined interrupt?
      01D7 816      BNEQ      REINIT_PORT              ; Branch if so.
      7E 50 7D 01D9 817      MOVQ      R0,-(SP)         ; Save more registers
54 0084 C5 D0 01DC 818      MOVL      UCB$$_PDT(R5),R4    ; Get PDT address
      01E1 819      $DEBUGCHECK #ERR$V_DEB_PSRX        ; Optionally bugcheck
      54 63 D0 01F4 820      MOVL      IDB$$_CSR(R3),R4    ; Retrieve config register
      01F7 821      ; address if bugcheck disabled
50 8005 8F 32 01F7 822      CVTWL     #<PAERSK_ES_UXIN ! ^X8000>, R0 ; Setup and log
      2A 11 01FC 823      BRB      HDWR_ERR_CODE        ; an unexpected interrupt.
      01FE 824
      01FE 825      OTHER_CNF_ERR:
      01FE 826
52 001EE700 8F D3 01FE 827      BITL      #FATAL_CNFERR,R2 ; Fatal error bit in config reg?
      19 12 0205 828      BNEQ      REINIT_PORT        ; Branch if so
      39 52 17 E0 0207 829      BBS      #PA_CNF_V_PDN,R2,PWR_DN ; Branch if port power down
      58 52 16 E0 020B 830      BBS      #PA_CNF_V_PUP,R2,PWR_UP ; Branch if port power up
      8F 52 0C E1 020F 831      BBC      #PA_CNF_V_NOCI,R2,- ; Branch if NOCI not set either
      0213 832      CNF_OK ; go check elsewhere for error
      0213 833
      0213 834      NO_C1750:
      0213 835
50 7E 50 7D 0213 836      MOVQ      R0,-(SP)         ; Save more registers
      8002 8F 32 0216 837      CVTWL     #<PAERSK_ES_HWER ! ^X8000>,R0 ; Log a non-specific
      FDE2 30 021B 838      BSBW      ELOG$HARDWARE    ; hardware error.
      18 11 021E 839      BRB      UNRECOV_ERR        ; Join general port
      0220 840      ; reinit code, but with
      0220 841      ; no retries permitted.
      0220 842
      0220 843      REINIT_PORT:
```


PA_INTERRUPT_SERVICE ROUTINE

```

      0220      844
50  7E  50  7D  0220      845      MOVQ  R0,-(SP)      ; Save more registers
      8002  8F  32  0223      846      CVTWL  #<PAERSK_ES_HWER ! ^X8000>, R0 ; Log a non-specific error
      0228      847
      0228      848 HDWR_ERR_CODE:
      0228      849
      0228      850      BSBW  ELOG$HARDWARE      ; Log registers
04  A4  01  D0  0228      851      MOVL  #PA_PMC_M_MIN,PA_PMC(R4); Do a maint init on port in case
      022F      852      ; we are out of init retries
      51  2C  3C  022F      853      MOVZWL #SS$_ABORT,R1      ; Assume we are not out of retries,
      0232      854      ; but tell SYSAP not to expect
      0232      855      ; cached send dg's back
      0080  C5  97  0232      856      DECB  UCB$_ERTCNT(R5)      ; Decr retry count
      05  18  0236      857      BGEQ  10$      ; Branch if not out of retries
      0238      858
      0238      859 UNRECOV_ERR:
      0238      860
      0238      861      MOVZWL #SS$_CTRLERR,R1      ; Else set aux status to tell SYSAP's
      023D      862      ; port won't be coming back
      023D      863
54  0084  C5  D0  023D      864 10$: MOVL  UCB$_PDT(R5),R4      ; Get PDT addr
      18  11  0242      865      BRB    20$      ; Join common crashed port code
      0244      866
      0244      867 PWR_DN:
      0244      868
      7E  50  7D  0244      869      MOVQ  R0,-(SP)      ; Save more registers
      50  03  9A  0247      870      MOVZBL #PAERSK_ES_PDWN, R0      ; Log a power down error.
      FDB3' 30  024A      871      BSBW  ELOG$HARDWARE
54  0084  C5  D0  024D      872      MOVL  UCB$_PDT(R5),R4      ; Get PDT addr
      02  AA  0252      873      BICW2  #PDT$_PUP,-      ; Clear PUP in PDT
      0110  C4  0254      874      PDTSW [PORT_STS(R4)
51  0364  8F  3C  0257      875      MOVZWL #SS$_POWERFAIL,R1      ; Set aux status to report to SYSAP's
      025C      876
      00  E2  025C      877 20$: BBSS  #PDT$_PWF_CLNUP,-      ; Set cleanup in progress
      0110  C4  025E      878      PDTSW [PORT_STS(R4),-      ; Dismiss interrupt if SYSAP
      37  0261      879      DISMISS_ERR_INT      ; notification is already in progress
      FD9B' 30  0262      880      BSBW  ERR$_PWF_RECOV      ; Call routine to set unit offline,
      0265      881      ; and fork to notify SYSAP's to
      0265      882      ; DISCONNECT
      32  11  0265      883      BRB    DISMISS_ERR_INT      ; Go dismiss interrupt
      0267      884
      0267      885 PWR_UP:
      0267      886
      7E  50  7D  0267      887      MOVQ  R0,-(SP)      ; Save more registers
      50  04  9A  026A      888      MOVZBL #PAERSK_ES_PUP, R0      ; Log a power up error.
      FD90' 30  026D      889      BSBW  ELOG$HARDWARE
      04  CA  0270      890      BICL  #PA_PMC_M_MIE,-      ; Disable interrupts on CI port,
      04  A4  0272      891      PA_PMC(R4)      ; but leave pup set to show CI
      0274      892      ; has powered up again
54  0084  C5  D0  0274      893      MOVL  UCB$_PDT(R5),R4      ; Get PDT addr
      1E  13  0279      894      BEQL  DISMISS_ERR_INT      ; Branch if port shut down and PDT gone
      02  A8  027B      895      BISW  #PDT$_PUP,-      ; Set power up occurred
      0110  C4  027D      896      PDTSW [PORT_STS(R4)      ; in port status
      00  E0  0280      897      BBS    #PDT$_PWF_CLNUP,-      ; Branch if pwr failure cleanup
      0110  C4  0282      898      PDTSW [PORT_STS(R4),-      ; still underway
      13  0285      899      DISMISS_ERR_INT      ;
      0286      900
```


PA_INTERRUPT_SERVICE ROUTINE

```

54 00E4 C4 D0 0286 901      MOVL   PDT$LCNF(R4),R4      ; Get config register addr
    04 A4 01 D0 028B 902      MOVL   #PA_PMC_M_MIN,PA_PMC(R4); Place port in un-initialized state
53 00000000'EF 9E 028F 903      MOVAB  INISPORT,R3      ; Re-initialization routine address
    FD67' 30 0296 904      BSBW    INISFORK      ; Create the fork process to perform
    0299 905      ; the port re-initialization
    0299 906
    0299 907 DISMISS_ERR_INT:
    0299 908
    3F BA 0299 909      POPR    #^M<R0,R1,R2,R3,R4,R5> ; Restore full complement of registers
    02 029B 910      REI      ; Exit interrupt

```

HANDLE_INT, HANDLE PORT INTERRUPT

```

029C 912 .SBTTL HANDLE_INT, HANDLE PORT INTERRUPT
029C 913 :+
029C 914 : This routine forks immediately leaving the address of the port
029C 915 : UCB0 in R5. From the UCB, the PDT is obtained.
029C 916 :
029C 917 : Entry REM_RSP is where a response is removed from the response
029C 918 : queue and checked for error. Packet errors are handled
029C 919 : in RSP_ERROR and other routines labelled 'RSP_...'.
029C 920 :
029C 921 : Error-free dequeued responses are dispatched on the message opcode.
029C 922 : Note that the 'message received' opcode is optimized by checking
029C 923 : for it first before casing on other opcodes. Unrecognized
029C 924 : opcodes are handled by optionally bugchecking or if the bugcheck
029C 925 : is disabled by crashing and reinitializing the port.
029C 926 :
029C 927 : Legal opcodes are handled by branching to their handlers. Handler
029C 928 : routine names are of the form REC_'opcode'.
029C 929 :
029C 930 : The message is processed by the handler which completes with a
029C 931 : branch back to REM_NEXT_RSP to dequeue the next response. Thus
029C 932 : a single interrupt results in several responses being processed
029C 933 : if they are in the queue. Responses are dequeued and processed
029C 934 : until none remain on the queue. At this time the I/O fork process
029C 935 : completes with an RSB.
029C 936 :
029C 937 : As a performance optimization, REM_NEXT_RSP first checks for
029C 938 : an empty response queue header with a TSTL rather than trying
029C 939 : to always attempting the REMQHI. This optimization is worth
029C 940 : doing since it is actually relatively rare that multiple responses
029C 941 : pile up on the response queue.
029C 942 :
029C 943 : Inputs:
029C 944 :
029C 945 : R4 -Addr of port configuration register
029C 946 : R5 -Addr of port UCB, unit 0
029C 947 :-
029C 948 :
029C 949 .ENABL LSB
029C 950
029C 951 HANDLE_INT:
029C 952
029C 953 IOFORK : Fork and lower IPL
029C 954 BICW #UCB_M_FKLOCK,- : Clear fork block in use
029C 955 UCBSQ_DEVSTS(R5) : lock bit
029C 956 MOVL UCBSL_PDT(R5),R4 : Get PDT address
029C 957
029C 958 REM_RSP: : Entry for removing a response
029C 959
029C 960 $REM_RESPQ : Get next response, addr in R2
029C 961 BVS NO_RSP : Branch if no more responses
029C 962 BBC #PPDSV_ERR,- : Branch if no error set
029C 963 PPDSB_STATUS(R2),- : in response status
029C 964 OPCODE_DISP :
029C 965 BRW RSP_ERROR : Else go handle error and
029C 966 : dispose of the response
029C 967 : Continue with known good
029C 968 : good response or ignorable error.

```

54 02 AA
68 A5
0084 C5 D0
18 1D
00 E1
0D A2
0A
01E7 31

HANDLE_INT, HANDLE PORT INTERRUPT

```

0200 C4 D5 02C9 969
DC 12 02C9 970 REM_NEXT_RSP:
05 02C9 971
02CD 972 TSTL PDT$Q_RSPQ(R4) ; Response queue empty?
02CF 973 BNEQ REM_RSP ; Branch if not
02D0 974 RSB ; Else exit fork
02D0 975
02D0 976 OPCODE_DISP:
02D0 977
OE A2 91 02D0 978 CMPB PPD$B_OPC(R2),- ; Is this a message receive?
22 02D3 979 #PPD$C_MSGREC ;
04 12 02D4 980 BNEQ 10$ ; Branch if not
OOC1 31 02D6 981 BRW REC_MSGREC ; Branch if so
02D9 982
02D9 983 NO_RSP: ; No more responses, end of
02D9 984 ; fork process
05 02D9 985 RSB
02DA 986
02DA 987 10$: ; Begin giant dispatch:
02DA 988 $DISPATCH -
02DA 989 PPD$B_OPC(R2),TYPE=B,- ; Dispatch on opcode
02DA 990 <-
02DA 991 <PPD$C_SNDDG, REC_SNDDG>,- ; Datagram sent
02DA 992 <PPD$C_SNDMSG, REC_SNDMSG>,- ; Message sent
02DA 993 <PPD$C_RETCNF, REC_RETCNF>,- ; Confirm returned
02DA 994 <PPD$C_REQDAT, REC_REQDAT>,- ; Data requested
02DA 995 <PPD$C_REQID, REC_REQID>,- ; ID requested
02DA 996 <PPD$C_SNDRST, REC_SNDRST>,- ; Reset sent
02DA 997 <PPD$C_SNDSTR, REC_SNDSTR>,- ; Start sent
02DA 998 <PPD$C_REQMDAT, REC_REQMDAT>,- ; Maint data requested
02DA 999 <PPD$C_SNDDAT, REC_SNDDAT>,- ; Data sent
02DA 1000 <PPD$C_RETDAT, REC_RETDAT>,- ; Data returned
02DA 1001 <PPD$C_SNDMDAT, REC_SNDMDAT>,- ; Maint data sent
02DA 1002 <PPD$C_SETCKT, REC_SETCKT>,- ; Virtual circuit opened
02DA 1003 <PPD$C_RDCNT, REC_RDCNT>,- ; Counters read
02DA 1004 <PPD$C_DGREC, REC_DGREC>,- ; Datagram received
02DA 1005 <PPD$C_CNFREC, REC_CNFREC>,- ; Confirm received
02DA 1006 <PPD$C_MCNFREC, REC_MCNFREC>,- ; Maint confirm received
02DA 1007 <PPD$C_IDREC, REC_IDREC>,- ; Received ID
02DA 1008 <PPD$C_DATREC, REC_DATREC>,- ; Data received
02DA 1009 <PPD$C_MDATREC, REC_MDATREC>,- ; Maint data received
02DA 1010 <PPD$C_SNDLB, REC_SNDLB>,- ; Loopback dg sent
02DA 1011 <PPD$C_LBREC, REC_LBREC>,- ; Loopback dg received
02DA 1012 <PPD$C_INVTC, REC_INVTC>,- ; Invalidate xlation cache
02DA 1013 >
0345 1014 ; End of giant dispatch
0345 1015 INV_OPCODE: ; Fall through to undefined
0345 1016 ; opcode
0345 1017
0345 1018 ;
0345 1019 ; Messages not handled for one reason or another:
0345 1020 ;
0345 1021
0345 1022 REC_SNDDAT: ; No success response ever requested
0345 1023 REC_RETCNF: ; No success response ever requested
0345 1024 REC_REQDAT: ; No success response ever requested
0345 1025 REC_REQMDAT: ; Command never issued

```

HANDLE_INT, HANDLE PORT INTERRUPT

		0345	1026	REC_RETDAT:		; No success response ever requested
		0345	1027	REC_SNDMDAT:		; Command never issued
		0345	1028	REC_MCNFREC:		; Command never issued
		0345	1029	REC_MDATREC:		; Command never issued
		0345	1030	REC_SNDLB:		; No success response ever requested
		0345	1031			
51	D4	0345	1032	\$DEBUGCHECK #ERRSV_DEB_INVOP		; Optional bugcheck
		0358	1033	CLRL R1		; Show no path available (response
		035A	1034			; isn't trustworthy enough to use
		035A	1035			; in PB lookup)
02C2	31	035A	1036	BRW RSP_CRASH_PORT		; Go init port crash
		035D	1037			
		035D	1038	.DSABL LSB		


```

035D 1040      .SBTTL HANDLERS FOR RESPONSES WITH GOOD STATUS
035D 1041      .SBTTL -      REC_CNFREC,      SEND DATA IS COMPLETE
035D 1042      .SBTTL -      REC_DATREC,      REQUEST DATA IS COMPLETE
035D 1043
035D 1044      :+
035D 1045      : These routines perform the same steps.
035D 1046      :
035D 1047      : First, the CONID portion of the XCT_ID is verified and converted
035D 1048      : to CDT address. The RSPID portion of the XCT_ID is converted to the
035D 1049      : response descriptor address and the CDRP address extracted from the
035D 1050      : RD. The RSPID and message buffer containing the CNFREC/DATREC
035D 1051      : are then deallocated. Finally, the context of the suspended SYSAP
035D 1052      : is restored and the SYSAP called back at the PC following the
035D 1053      : call to send/request data.
035D 1054      :
035D 1055      : Inputs:
035D 1056      :
035D 1057      :      R2      -Addr of message
035D 1058      :      R4      -Addr of PDT
035D 1059      :
035D 1060      : Outputs:
035D 1061      :
035D 1062      :      R0-R3      -Destroyed
035D 1063      :      Other registers      -Preserved
035D 1064      :-
035D 1065      :
035D 1066      :      .ENABL  LSB
035D 1067      :
035D 1068      :      REC_DATREC:
035D 1069      :
035D 1070      :      REC_CNFREC:
035D 1071      :
035D 1072      :      .IF      DF PA$DEBUG      ; Debug facility
035D 1073      :      BSBW      TRC$LOGMSG      ; Log message
035D 1074      :      .ENDC
035D 1075      :
035D 1076      :      ADDL      PDT$ MSGHDRSZ(R4),R2      ; Compute addr of application data
035D 1077      :      BSBW      FPC$REC_CNFREC      ; Go process it
035D 1078      :      BRW      REM_NEXT_RSP      ; Get next pkt from response queue
035D 1079      :
035D 1080      :      .DSABL  LSB

```

52 00B4 C4 C0
 FC9B' 30
 FF61 31

- REC_DGREC, PROCESS RECEIVED DG

```

0368 1082 .SBTTL - REC_DGREC, PROCESS RECEIVED DG
0368 1083
0368 1084 ;+
0368 1085 ; If the PPD type of the received datagram is not SCS_DG, then the datagram
0368 1086 ; is assumed to be a start handshake datagram and is given to the CONFIG
0368 1087 ; module, routine CNF$DGREC, to process. Otherwise, REC_DGREC verifies
0368 1088 ; the destination connection ID and checks that the connection has at
0368 1089 ; least one datagram queued for receive. If the connection has no datagrams
0368 1090 ; queued for receive, then the datagram is discarded to the free queue and not
0368 1091 ; given to the SYSAP. Otherwise, the SYSAP's datagram input address is called.
0368 1092 ; Upon return from the SYSAP, branch is taken to REM_NEXT_RSP to get the
0368 1093 ; next response.
0368 1094 ;
0368 1095 ; Inputs:
0368 1096 ;
0368 1097 ; R2 -Addr of message
0368 1098 ; R4 -Addr of PDT
0368 1099 ;
0368 1100 ; Outputs:
0368 1101 ;
0368 1102 ; R1 -Length of application data
0368 1103 ; R0,R2,R3,R5 -Destroyed
0368 1104 ; Other registers -Preserved
0368 1105 ;-
0368 1106
0368 1107 ASSUME SYSAP$C_DGREC EQ 0
0368 1108
0368 1109 .ENABL LSB
0368 1110
0368 1111 REC_DGREC:
0368 1112
0368 1113 .IF DF PAS$DEBUG ; Debug facility
0368 1114 BSBW TRC$LOGMSG ; Log datagram
0368 1115 .ENDC ;
0368 1116
12 A2 B1 0368 1117 CMPW PPD$W_MTYPE(R2),- ; Is PPD msg type = SCS?
03 03 0368 1118 #PPD$C_SCS_DG ;
06 13 0368 1119 BEQL 10$ ; Branch if so
FC8F' 30 0368 1120 BSBW CNF$DGREC ; Else pass msg to configuration
FF55 31 0371 1121 BRW REM_NEXT_RSP ; Get next response
0374 1122
52 0190 C4 C0 0374 1123 10$: ADDL PDT$L_DGHDRSZ(R4),R2 ; Compute addr of application data
FC84' 30 0379 1124 BSBW FPC$REC_DGREC ; Call SCS layer
FF4A 31 037C 1125 BRW REM_NEXT_RSP ; Get next response
037F 1126
037F 1127 .DSABL LSB

```


- REC_IDREC, PROCESS RECEIVED ID

```

037F 1129 .SBTTL - REC_IDREC, PROCESS RECEIVED ID
037F 1130
037F 1131 ;+
037F 1132 ; If the transaction ID = 0, then this ID packet is either unsolicited
037F 1133 ; or a response to the configuration poller and it is given to the CONFIG
037F 1134 ; module, entry CNF$IDREC.
037F 1135 ;
037F 1136 ;
037F 1137 ; Inputs:
037F 1138 ;
037F 1139 ; R2 -Addr of message
037F 1140 ; R4 -Addr of PDT
037F 1141 ;
037F 1142 ; Outputs:
037F 1143 ;
037F 1144 ; R0-R3,R5 -Destroyed
037F 1145 ; Other registers -Preserved
037F 1146 ; -
037F 1147 ;
037F 1148 .ENABL LSB
037F 1149
037F 1150 REC_IDREC:
037F 1151
10 A2 D5 037F 1152 TSTL PPD$Q_XCT_ID(R2) ; Check l.o. XCT_ID
0A 12 0382 1153 BNEQ 10$ ; Branch if non-zero
14 A2 D5 0384 1154 TSTL PPD$Q_XCT_ID+4(R2) ; Check h.o. XCT_ID
05 12 0387 1155 BNEQ 10$ ; Branch if non-zero
FC74 30 0389 1156 BSBW CNF$IDREC ; Else call config module
03 11 038C 1157 BRB 20$ ; Get next response
038E 1158
0463 30 038E 1159 10$: BSBW INT$INS_DFREQ1 ; Return buffer to free
0391 1160 ; queue and ignore
0391 1161
FF35 31 0391 1162 20$: BRW REM_NEXT_RSP ; Get next response
0394 1163
0394 1164 .DSABL LSB

```

- REC_LBREC, PROCESS RECEIVED LB DG

0394 1166 .SBTTL - REC_LBREC, PROCESS RECEIVED LB DG

0394 1167
0394 1168 :+ The loopback datagram is verified and discarded by CNF\$LBREC
0394 1169 :
0394 1170 :
0394 1171 : Inputs:

0394 1172 :
0394 1173 : R2 -Addr of LB dg
0394 1174 : R4 -Addr of PDT
0394 1175 :
0394 1176 : Outputs:

0394 1177 :
0394 1178 : R0-R2 -Destroyed
0394 1179 :
0394 1180 : Other registers -Preserved
0394 1181 :-

0394 1182 :
0394 1183 : .ENABL LSB
0394 1184 :
0394 1185 REC_LBREC:

FC69' 30 0394 1187 BSBW CNF\$LBREC ; Call routine in PACONFIG
FF2F 31 0397 1188 BRW REM_NEXT_RSP ; Get next response
039A 1189
039A 1190 .DSABL LSB

- REC_MSGREC, PROCESS RECEIVED MESSAGE

```

039A 1192      .SBTTL -      REC_MSGREC,      PROCESS RECEIVED MESSAGE
039A 1193
039A 1194      :+
039A 1195      : REC_MSGREC checks the SCS message type field.  If the type code
039A 1196      : is SCSSC_APPL_MSG, then processing continues.  Otherwise, the message
039A 1197      : is an SCS control message and routine SCSSREC_SCSMSG in module PASCCTL
039A 1198      : is called.
039A 1199
039A 1200      : For application messages, REC_MSGREC checks that the connection
039A 1201      : ID is legal.  If not, the message buffer is discarded (returned to
039A 1202      : the free queue) and processing ends.  Otherwise, the connection credit
039A 1203      : bookkeeping is done and the SYSAP's message input address is called.
039A 1204      : The SYSAP is responsible for disposing of the message buffer.  Upon
039A 1205      : return from the SYSAP, REC_MSGREC branches to REM_NEXT_RSP.
039A 1206
039A 1207      : Inputs:
039A 1208
039A 1209      :      R2      -Addr of message
039A 1210      :      R4      -Addr of PDT
039A 1211      :      R5      -Addr of UCB 0
039A 1212
039A 1213      : Outputs:
039A 1214
039A 1215      :      R0-R3,R5      -Destroyed
039A 1216      :      Other registers      -Preserved
039A 1217      :-
039A 1218
039A 1219      .ENABL  LSB
039A 1220
039A 1221  REC_MSGREC:
039A 1222
039A 1223      .IF      DF PA$DEBUG      : Debug facility
039A 1224      BSBW      TRC$LOGMSG      : Log message
039A 1225      .ENDC
039A 1226
039A 1227      ADDL      PDT$L_MSGHDRSZ(R4),R2      : Compute addr of application data
039A 1228      BSBW      FPC$REC_MSGREC      : Call SCS layer
039A 1229      BRW      REM_NEXT_RSP      : Get next response
039A 1230
039A 1231      .DSABL  LSB

```

52 00B4 C4 C0
FC5E' 30
FF24 31

- REC_RDCNT, PORT COUNTERS READ

```

03A5 1233 .SBTTL - REC_RDCNT, PORT COUNTERS READ
03A5 1234
03A5 1235 :+
03A5 1236 : REC_RDCNT returns the received buffer of port counters to the
03A5 1237 : SYSAP that owns the port counters currently. If the SYSAP specified
03A5 1238 : a release of the counters, then the counters busy flag is cleared.
03A5 1239 :
03A5 1240 : Inputs:
03A5 1241 :
03A5 1242 : R2 -Addr of dg buffer containing counters
03A5 1243 : R4 -Addr of PDT
03A5 1244 :
03A5 1245 : PDT$L_CNTCDRP(R4) -CDRP holding suspended SYSAP context
03A5 1246 :
03A5 1247 : Outputs:
03A5 1248 :
03A5 1249 : R0-R3,R5 -Destroyed
03A5 1250 :
03A5 1251 : Other registers -Preserved
03A5 1252 :
03A5 1253 : PDT$W_FLAGS(R4) -If PDT$M_CNTRLs is set, then PDT$M_CNTRLs
03A5 1254 : and PDT$M_CNTBSY are both cleared
03A5 1255 :-
03A5 1256 :
03A5 1257 : .ENABL LSB
03A5 1258 :
03A5 1259 REC_RDCNT:
03A5 1260
FC58' 30 03A5 1261 BSBW FPC$REC_RDCNT ; Call SCS layer
FF1E' 31 03A8 1262 BRW REM_NEXT_RSP ; Go for next response
03AB 1263
03AB 1264 .DSABL LSB

```


- REC_SETCKT, SET CIRCUIT DONE

.SBTTL - REC_SETCKT, SET CIRCUIT DONE

03AB 1266
03AB 1267
03AB 1268
03AB 1269
03AB 1270
03AB 1271
03AB 1272
03AB 1273
03AB 1274
03AB 1275
03AB 1276
03AB 1277
03AB 1278
03AB 1279
03AB 1280
03AB 1281
03AB 1282
03AB 1283
03AB 1284
03AB 1285
03AB 1286
03AB 1287
03AB 1288
03AB 1289
03AB 1290
03AB 1291
03AB 1292
03AB 1293
03AB 1294
03AB 1295
03AB 1296
03AB 1297
03AB 1298
03AB 1299
03AB 1300
03AB 1301
03AD 1302
03B0 1303
03B3 1304
03B8 1305
03B8 1306
03BA 1307
03BC 1308
03C0 1309
03C3 1310
03C6 1311
03C6 1312
03C9 1313
03CC 1314
03CC 1315

:+ The only SETCKT issued for which a success notification is requested is in ERR\$CRASHVC which closes a VC on which a software-detected error has occurred. REC_SETCKT calls SC\$CLOSEDC to continue the failure process of notifying all SYSAP's with connections on the failing circuit and possibly deleting the path block if the SYSAP's all DISCONNECT promptly. All other SETCKT's with RETFLAG = TRUE are intended to simply return the datagram buffer to pool.

Inputs:

R2 -Addr of SETCKT dg
R4 -Addr of PDT
R5 -Addr of UCB 0

Outputs:

R0,R1 -Destroyed
Other registers -Preserved
PB\$CLCKT_DG in PB -Set to address of SETCKT dg to show dg belongs to PB again

.ENABL LSB

REC_SETCKT:

.IF DF PA\$DEBUG ; Debug facility
BSBW TRC\$LOGMSG ; Log set circuit
.ENDC ;

BBC #PPD\$V DISPOSE,- ; If notification not requested,
PPD\$B SWFLAG(R2),10\$; branch around it
BSBW CNF\$LRP PB MSG ; Go lookup PB from message
MOVZWL #SS\$VCCLOSED,R3 ; Assume aux status will be closed
 ; due to other than host shutdown
TSTL R1 ; Is there a PB?
BEQL 10\$; Branch if not
MOVL R2,PB\$CLCKT_DG(R1) ; Else put SETCKT dg address back in PB
BSBW SC\$SETCKT_CLSD ; Notify all SYSAP's with connections
BRW REM_NEXT_RSP ; Go for next response

10\$: BSBW INT\$DEAL_DG1 ; Return dg buffer to pool
BRW REM_NEXT_RSP ; Go for next response

.DSABL LSB

00 E1
16 0B A2
FC4D' 30
53 21A4 8F 3C
51 D5
0A 13
54 A1 52 D0
FC3D' 30
FF03 31
04E1 30
FEFD 31

- REC_SNDDG, PROCESS SENT DATAGRAM

```
03CC 1317      .SBTTL -      REC SNDDG,      PROCESS SENT DATAGRAM
03CC 1318      .SBTTL -      INT$DISP_SENDDG,DISPATCH A SENT DG
03CC 1319
03CC 1320      ;+
03CC 1321      ; REC SNDDG checks the datagram disposal flag, PPD$M_DISPOSE in
03CC 1322      ; PPD$B_SWFLAG. If the flag is 0, then the datagram buffer is
03CC 1323      ; deallocated to nonpaged pool. If the flag is set, then the sent
03CC 1324      ; datagram is passed to the SYSAP with R0 set to indicated that
03CC 1325      ; the datagram is a sent dg rather than a new received dg.
03CC 1326
03CC 1327      Inputs:
03CC 1328
03CC 1329      R2      -Addr of message
03CC 1330      R4      -Addr of PDT
03CC 1331
03CC 1332      Outputs:
03CC 1333
03CC 1334      R1      -Length of application data
03CC 1335      R0,R2,R3,R5 -Destroyed
03CC 1336      Other registers -Preserved
03CC 1337      :-
03CC 1338
03CC 1339      .ENABL  LSB
03CC 1340
03CC 1341      REC_SNDDG:
03CC 1342
03CC 1343      .IF      DF PA$DEBUG      ; Debug facility
03CC 1344      BSBW      TRC$LOGMSG      ; Log datagram
03CC 1345      .ENDC
03CC 1346
03CC 1347      BSBB      INT$DISP_SENDDG      ; Call routine to dispatch
03CE 1348      ; datagram
FEF8 31 03CE 1349      BRW      REM_NEXT_RSP      ; Go get next response
03D1 1350
03D1 1351
03D1 1352      INT$DISP_SENDDG::
03D1 1353
03D1 1354      CMPW      PPD$W_MTYPE(R2),-      ; Is PPD type = SCS?
03D4 1355      #PPD$C_SCS_DG      ; (If not, PPD dg is assumed)
03D5 1356      BNEQ      20$      ; Branch if not since returned
03D7 1357      ; PPD dg's always go back to pool
03D7 1358      BITB      #PPD$M_DISPOSE,-      ; Is disposal flag set?
03D9 1359      PPD$B_SWFLAG(R2)      ;
03DB 1360      BEQL      20$      ; Branch if clear, dg --> pool
52 0190 C4 C0 03DD 1361      ADDL      PDT$L_DGHDRSZ(R4),R2      ; Compute addr of appliation data
FC1B' 31 03E2 1362      BRW      FPC$REC_SNDDG      ; Call SCS layer
03E5 1363
03E5 1364      20$:      BBC      #PPD$V_RSP,-      ; Branch if this SNDDG was supposed
03E7 1365      PPD$B_FLAGS(R2),25$      ; to be returned to the dg free queue
04 0F A2 30 03EA 1366      BSBW      INT$DEAL_DG1      ; Else deallocate dg to pool
04BD 05 03ED 1367      RSB      ; Return
03EE 1368
0403 31 03EE 1369      25$:      BRW      INT$INS_DFREQ1      ; Insert back on dg free queue and RSB
03F1 1370
03F1 1371      .DSABL  LSB
```


- REC_SNDMSG, PROCESS SENT MESSAGE

```
03F1 1373 .SBTTL - REC_SNDMSG, PROCESS SENT MESSAGE
03F1 1374
03F1 1375 ;+
03F1 1376 : REC_SNDMSG simply calls FPC$DEALLMSG2 to deallocate the sent message.
03F1 1377 : The deallocate takes care of flow control and may deallocate the
03F1 1378 : buffer to the free queue if the free queue is low, or to pool.
03F1 1379 :
03F1 1380 : Inputs:
03F1 1381 :
03F1 1382 : R2 -Addr of message
03F1 1383 : R4 -Addr of PDT
03F1 1384 : R5 -Addr of UCB 0
03F1 1385 :
03F1 1386 : Outputs:
03F1 1387 :
03F1 1388 : R0-R3,R5 -Destroyed
03F1 1389 : Other registers -Preserved
03F1 1390 :-
03F1 1391
03F1 1392 .ENABL LSB
03F1 1393
03F1 1394 REC_SNDMSG:
03F1 1395
03F1 1396 .IF DF PA$DEBUG ; Debug facility
03F1 1397 BSBW TRC$LOGMSG ; Log message
03F1 1398 .ENDC ;
03F1 1399
52 00B4 C4 C0 03F1 1400 ADDL PDT$ MSGHDRSZ(R4),R2 ; Compute addr of application data
FC07' 30 03F6 1401 BSBW FPC$REC_SNDMSG ; Call SCS layer
FECD 31 03F9 1402 BRW REM_NEXT_RSP ; Get next response
03FC 1403
03FC 1404 .DSABL LSB
```

- REC_REQID, ID REQUESTED

03FC	1406	.SBTTL	-	REC_REQID,	ID REQUESTED
03FC	1407	.SBTTL	-	REC_SNDRST,	RESET SENT
03FC	1408	.SBTTL	-	REC_SNDSTRT	START SENT
03FC	1409				
03FC	1410	:	+		
03FC	1411	:		Request ID's sent with RETFLAG = TRUE come here. They are simply	
03FC	1412	:		deallocated to nonpaged pool. It is immaterial whether the REQID	
03FC	1413	:		packet incurred an error or not.	
03FC	1414	:			
03FC	1415	:		Maintenance reset and start are always sent with RETFLAG = TRUE in	
03FC	1416	:		order that their buffers can be reclaimed for pool.	
03FC	1417	:			
03FC	1418	:		Inputs:	
03FC	1419	:			
03FC	1420	:		R2	-Addr of datagram
03FC	1421	:		R4	-Addr of PDT
03FC	1422	:		R5	-Addr of UCB 0
03FC	1423	:			
03FC	1424	:		Outputs:	
03FC	1425	:			
03FC	1426	:		R0	-Destroyed
03FC	1427	:		Other registers	-Preserved
03FC	1428	:	-		
03FC	1429	:			
03FC	1430	:		.ENABL	LSB
03FC	1431	:			
03FC	1432	:		REC_INVTC:	
03FC	1433	:		REC_REQID:	
03FC	1434	:		REC_SNDRST:	
03FC	1435	:		REC_SNDSTRT:	
03FC	1436	:			
04AB	30	:		BSBW	INT\$DEAL_DG1 ; Return buffer to pool
FEC7	31	:		BRW	REM_NEXT_RSP ; Get next response
0402	1439	:			
0402	1440	:		.DSABL	LSB


```

0402 1442      .SBTTL PROCESSING OF ERROR STATUS IN RESPONSE
0402 1443
0402 1444      :+
0402 1445      : Branch to RSP_ERROR on all types of response status error, i.e.,
0402 1446      : PPD$M_ERR not 0. There are several error strategies depending
0402 1447      : upon the severity of the error and on whose fault it is most likely
0402 1448      : to be:
0402 1449
0402 1450      : -Crash the VC if this is a new error on this VC deemed to be
0402 1451      : the fault of the remote system or port.
0402 1452
0402 1453      : -Drain (or process) the response if the VC closure has already been
0402 1454      : initiated, e.g., if the status is 'virtual circuit closed.'
0402 1455
0402 1456      : -Bugcheck if this is an error can only be the fault of the local
0402 1457      : software.
0402 1458
0402 1459      : -Crash (reinit) the port and all VC's on it by simulation of a
0402 1460      : power fail recovery if this might be a local port failure or if
0402 1461      : this is a potential transient software failure.
0402 1462
0402 1463      : Dispatch first on the status type field as follows:
0402 1464
0402 1465      : Type value      Action
0402 1466
0402 1467      : PPD$C_TYPOK      Go to PATH_FAIL. Only one path
0402 1468      :                   failed, the transmission succeeded,
0402 1469      :                   so PATH_FAIL has no effect on the VC.
0402 1470
0402 1471      : PPD$C_TYPVCC      Virtual circuit closed due to
0402 1472      :                   previously reported error. In this
0402 1473      :                   case, go to RSP_VC_CLOSED to dispatch
0402 1474      :                   on opcode/ppd type code.
0402 1475
0402 1476      : PPD$C_TYPINVBN     Invalid buffer name. Crash port.
0402 1477
0402 1478      : PPD$C_TYPBVLV     Local buffer length violation.
0402 1479      :                   Crash port.
0402 1480
0402 1481      : PPD$C_TYPACCV     Block xfer local access control
0402 1482      :                   violation. Crash port.
0402 1483
0402 1484      : PPD$C_TYPPNP      No path left. Go to RSP_NO_PATH
0402 1485      :                   to dispatch on opcode/ppd type.
0402 1486
0402 1487      : PPD$C_TYPPMSE     Buffer memory system error.
0402 1488      :                   Port closed virtual circuit when it
0402 1489      :                   detected this error. Treat exactly
0402 1490      :                   like no path detected during block
0402 1491      :                   xfer and dispatch to RSP_CLOSED_VC.
0402 1492
0402 1493      : PPD$C_TYPOTHER     Other error defined by the subtype.
0402 1494      :                   Go to RSP_SUBTYP_CHK to dispatch
0402 1495      :                   on subtype code.
0402 1496
0402 1497
0402 1498      : At RSP_SUBTYP, the subtype error status is picked up and dispatched on

```

```

0402 1499 : as follows:
0402 1500 :
0402 1501 : Subtype code Action
0402 1502 :
0402 1503 : PPD$C_STPSV Pkt size violation. Go to RSP_PKTSTZ_VIO
0402 1504 : to dispatch on opcode and ppd type
0402 1505 : since some opcodes represent errors
0402 1506 : generated at the remote side and some
0402 1507 : represent local errors.
0402 1508 :
0402 1509 : PPD$C_STURP Unrecognized packet. These are
0402 1510 : packets with invalid information
0402 1511 : received from remote systems. Go
0402 1512 : to RESP_UNREC_PKT to log and discard.
0402 1513 :
0402 1514 : PPD$C_STINVDP Invalid destination port. Crash port.
0402 1515 :
0402 1516 : PPD$C_STURC Unrecognized local command. Crash port.
0402 1517 :
0402 1518 : PPD$C_STABO Aborted command. This status is returned
0402 1519 : upon an orderly (host requested) disable
0402 1520 : of the port. Since this disable is currently
0402 1521 : never requested by PADRIVER, this status is
0402 1522 : illegal and causes a port crash.
0402 1523 :
0402 1524 : Inputs:
0402 1525 :
0402 1526 : R2 -Addr of response pkt
0402 1527 : R4 -PDT addr
0402 1528 :
0402 1529 : Outputs:
0402 1530 :
0402 1531 : R4 -Preserved
0402 1532 :
0402 1533 : Branch back to REM_NEXT_RSP
0402 1534 : unless port is crashed in which
0402 1535 : case return is taken to the fork
0402 1536 : dispatcher.
0402 1537 :-
0402 1538 :

```


- MACROS TO DEFINE ACTION TABLE

```

0402 1540      .SBTTL -      MACROS TO DEFINE ACTION TABLE
0402 1541
0402 1542      :+
0402 1543      : Each status requiring opcode/ppd type specific action has a
0402 1544      : table of opcodes, PPD types, and an action routine to call for
0402 1545      : each opcode-ppd type combination. The format of the table is as
0402 1546      : follows:
0402 1547
0402 1548      : .BYTE opcode
0402 1549      : .BYTE link to next opcode, 0 if no more
0402 1550      : .WORD  ppd type, -1 if any ppd type ok
0402 1551      : .WORD  offset to action routine
0402 1552      : ...
0402 1553
0402 1554      : .BYTE opcode
0402 1555      : ...
0402 1556
0402 1557      : Note that for each opcode-ppd type pair, only one action routine
0402 1558      : is allowed. There may be several possible ppd types for a given
0402 1559      : opcode, each specifying a different action routine, but the list
0402 1560      : of ppd types should end with ppd type of OTHER to handle all other
0402 1561      : values of ppd type.
0402 1562
0402 1563      : The following macros define entries in the action tables:
0402 1564
0402 1565      : OPCODE opcode,[rtn],[last]      ; Defines the opcode byte and
0402 1566      :                                     ; link to next opcode. Link
0402 1567      :                                     ; is -1 if last not specified.
0402 1568      :                                     ; If rtn is specified, then
0402 1569      :                                     ; PPDTYPE ANY, and ACTION RTN
0402 1570      :                                     ; are also invoked.
0402 1571
0402 1572      : PPDTYPE typc      ; Defines the ppd type field,
0402 1573      :                                     ; -1 if typc = ANY or OTHER.
0402 1574
0402 1575      : ACTION rtn      ; Defines the offset to the
0402 1576      :                                     ; action routine relative to
0402 1577      :                                     ; the ppd type code.
0402 1578      : -
0402 1579
0402 1580      : Define OPCODE opcode,[rtn],[last] macro:
0402 1581      :
0402 1582      :
0402 1583      : .MACRO OPCODE OPC,RTN,LAST
0402 1584
0402 1585      : .NOSHOW
0402 1586      : $$$=
0402 1587      : .BYTE PPD$C 'OPC'      ; Save start of this entry
0402 1588      :                                     ; Opcode
0402 1589      : .IF NE $$$LAST OPC      ; If there was a previous
0402 1590      : .=$$$LAST OPC+EOASB NEXTOPC      ; opcode entry, go back and
0402 1591      : .BYTE $$$=$$$LAST OPC      ; fill in its fwd link
0402 1592      : .=$$$+EOASB_NEXTOPC      ; and reset ptr to this entry
0402 1593      : .ENDC
0402 1594
0402 1595      : .BYTE 0      ; Set fwd link to no more for now
0402 1596

```

- MACROS TO DEFINE ACTION TABLE

```
0402 1597      $$$LAST_OPC=$$$      ; Redefine last opcode to this
0402 1598
0402 1599      .IF      NB LAST      ; If this is last opcode in
0402 1600      $$$LAST_OPC=0      ; in action table, then reset
0402 1601      .ENDC      ; last opcode to 0
0402 1602
0402 1603      .IF      NB RTN      ; If action routine specified,
0402 1604      PPDTYP ANY      ; the set ppd type to any ok
0402 1605      ACTION RTN      ; and fill in action routine offset
0402 1606      .ENDC      ;
0402 1607      .SHOW
0402 1608
0402 1609      .ENDM      OPCODE
0402 1610
00000000 0402 1611 $$$LAST_OPC = 0      ; Initialize addr of last opcode entry
0402 1612
0402 1613      :
0402 1614      : Define macro to specify PPD type:
0402 1615      :
0402 1616
0402 1617      .MACRO PPDTYP TYPC
0402 1618
0402 1619      .NOSHOW
0402 1620      $$$=.      ; Save start of this entry
0402 1621      .IF      IDN TYPC,ANY      ; If any ppd type ok,
0402 1622      .WORD      -1      ; set ppd type to -1
0402 1623      .ENDC
0402 1624      .IF      IDN TYPC,OTHER      ; If any ppd type ok,
0402 1625      .WORD      -1      ; set ppd type to -1
0402 1626      .ENDC
0402 1627      .IF      DIF TYPC,ANY      ; If neither any nor
0402 1628      .IF      DIF TYPC,OTHER      ; other is specified, then
0402 1629      .WORD      PPD$C_'TYPC'      ; set specified ppd type code
0402 1630      .ENDC
0402 1631      .ENDC
0402 1632      .SHOW
0402 1633
0402 1634      .ENDM      PPDTYP
0402 1635
0402 1636      :
0402 1637      : Define ACTION macro to define offset to action routine to call:
0402 1638      :
0402 1639
0402 1640      .MACRO ACTION RTN
0402 1641
0402 1642      .NOSHOW
0402 1643      .WORD      RTN-$$$      ; Offset = routine addr -
0402 1644      .SHOW      ; ppd type code addr
0402 1645
0402 1646      .ENDM      ACTION
0402 1647
0402 1648
0402 1649      :
0402 1650      : Define symbolic offsets to the opcode entry in the error-opcode
0402 1651      : action table (EOA) and to the ppd action entry (PPA):
0402 1652      :
0402 1653
```


- MACROS TO DEFINE ACTION TABLE

00000000	0402	1654	EOASB_OPC	=0	; Opcode
00000001	0402	1655	EOASB_NEXTOPC	=1	; Link to next opcode, 0 if none
00000002	0402	1656	EOASW_PPDTP	=2	; 1st ppd type code for this opcode
	0402	1657			
00000000	0402	1658	PPASW_PPDTP	=0	; PPD type code
00000002	0402	1659	PPASW_RTN	=2	; Action routine offset
00000004	0402	1660	PPASC_LENGTH	=4	; Length of PPD action entry

- OPCODE-DEPENDENT ERROR ACTION TABLE

```

0402 1662 .SBTTL - OPCODE-DEPENDENT ERROR ACTION TABLE
0402 1663
0402 1664 :
0402 1665 : Define the opcode/ppd type specific handling for the three
0402 1666 : kinds of error that require opcode examination:
0402 1667 :
0402 1668
0402 1669 ACT_NO_PATH: ; No path status
0402 1670
0402 1671 OPCODE SNDDG
0404 1672
0404 1673 PPDTYP SCS_DG
0406 1674 ACTION RSP_CRASH_VC
0408 1675
0408 1676 PPDTYP OTHER
040A 1677 ACTION RSP_IGNORE_ERR
040C 1678
040C 1679 OPCODE SNDMSG
040E 1680
040E 1681 PPDTYP SCS_MSG
0410 1682 ACTION RSP_CLOSED_VC
0412 1683
0412 1684 PPDTYP CACHECLR
0414 1685 ACTION RSP_CACHECLR
0416 1686
0416 1687 PPDTYP OTHER
0418 1688 ACTION RSP_CRASH_NPUPD
041A 1689
041A 1690 OPCODE SNDDAT, RTN=RSP_CLOSED_VC
0420 1691
0420 1692 OPCODE RETDAT, RTN=RSP_CLOSED_VC
0426 1693
0426 1694 OPCODE RETCNF, RTN=RSP_CLOSED_VC
042C 1695
042C 1696 OPCODE REQDAT, RTN=RSP_CLOSED_VC
0432 1697
0432 1698 OPCODE SNDLB, RTN=RSP_DISCARD_ERR
0438 1699
0438 1700 OPCODE REQID, RTN=RSP_PATH_FAIL
043E 1701
043E 1702 OPCODE SNDRST, RTN=RSP_CRASH_VC
0444 1703
0444 1704 OPCODE SNDSTRT, RTN=RSP_CRASH_VC, -
0444 1705 LAST=TRUE
044A 1706
044A 1707
044A 1708 ACT_PKTSIZ_VIO:
044A 1709
044A 1710 OPCODE SNDDG, RTN=RSP_CRASH_PSV
0450 1711
0450 1712 OPCODE SNDMSG, RTN=RSP_CRASH_PSV
0456 1713
0456 1714 OPCODE SNDDAT, RTN=RSP_CRASH_PSV
045C 1715
045C 1716 OPCODE DATREC, RTN=RSP_CRASH_PSV
0462 1717
0462 1718 OPCODE SNDLB, RTN=RSP_CRASH_PSV

```


0468	1719		
0468	1720	OPCODE	LBREC, RTN=RSP_CRASH_PSV
046E	1721		
046E	1722	OPCODE	MSGREC, RTN=RSP_CRASH_VC
0474	1723		
0474	1724	OPCODE	RETDAT, RTN=RSP_CRASH_VC
047A	1725		
047A	1726	OPCODE	RETCNF, RTN=RSP_CRASH_VC
0480	1727		
0480	1728	OPCODE	DGREC, LAST=TRUE
0482	1729		
0482	1730	PPDTYP	SCS_DG
0484	1731	ACTION	RSP_CRASH_VC
0486	1732		
0486	1733	PPDTYP	OTHER
0488	1734	ACTION	RSP_IGNORE_ERR
048A	1735		
048A	1736		
048A	1737		
048A	1738	ACT_VC_CLOSED:	: VC already closed status
048A	1739		
048A	1740	OPCODE	SNDDMSG
048C	1741		
048C	1742	PPDTYP	SCS_MSG
048E	1743	ACTION	RSP_DRAIN_ERR
0490	1744		
0490	1745	PPDTYP	CACHECLR
0492	1746	ACTION	RSP_CACHECLR
0494	1747		
0494	1748	PPDTYP	OTHER
0496	1749	ACTION	RSP_CRASH_VCUPD
0498	1750		
0498	1751	OPCODE	SNDDAT, RTN=RSP_DRAIN_ERR
049E	1752		
049E	1753	OPCODE	RETDAT, RTN=RSP_DRAIN_ERR
04A4	1754		
04A4	1755	OPCODE	RETCNF, RTN=RSP_DRAIN_ERR
04AA	1756		
04AA	1757	OPCODE	REQDAT, RTN=RSP_DRAIN_ERR,-
04AA	1758		LAST=TRUE

- RSP_ERROR, DISPATCH ON ERROR

```

04B0 1760 .SBTTL - RSP_ERROR, DISPATCH ON ERROR
04B0 1761 .SBTTL - TYPE
04B0 1762
04B0 1763 ;+
04B0 1764 ; RSP_ERROR is branched to when a response packet with any sort
04B0 1765 ; of error is dequeued. RSP_ERROR dispatches on the error type and
04B0 1766 ; subtype fields in the status.
04B0 1767 ;-
04B0 1768
04B0 1769 .ENABL LSB
04B0 1770
04B0 1771 RSP_ERROR:
04B0 1772
FB4D' 30 04B0 1773 BSBW CNF$CLKP_PB_MSG ; Get the path block in R1
04B3 1774 ; Ignore PB not found error
04B3 1775 ; since this will be checked
04B3 1776 ; later if needed
05 EF 04B3 1777 EXTZV #PPD$V_STSTYP,- ; Extract type code
03 04B5 1778 #PPD$S_STSTYP,- ; from response
50 OD A2 04B6 1779 PPD$B_STATUS(R2),R0
04B9 1780
04B9 1781 $DISPATCH R0,TYPE=B,- ; Dispatch on type code:
04B9 1782 <-
04B9 1783 <PPD$C_TYPOK, RSP_PATH_FAIL>,- ; A path failed
04B9 1784 <PPD$C_TYVCC, RSP_VC_CLOSED>,- ; VC closed
04B9 1785 <PPD$C_TYINVB, RSP_CRASH_INVB>,- ; Invalid buffer name
04B9 1786 <PPD$C_TYBLV, RSP_CRASH_BLV>,- ; Buffer length violation
04B9 1787 <PPD$C_TYACCV, RSP_CRASH_ACCV>,- ; Access control violation
04B9 1788 <PPD$C_TYPNP, RSP_NO_PATH>,- ; No path left
04B9 1789 <PPD$C_TYBMSE, RSP_VC_CLOSED>,- ; Buffer memory system error
04B9 1790 <PPD$C_TYOTHER, RSP_SUBTYP_CHK>,- ; Other error
04B9 1791 >
04CD 1792
04CD 1793 RSP_SUBTYP_CHK:
04CD 1794
01 EF 04CD 1795 EXTZV #PPD$V_STSST,- ; Extract subtype
04 04CF 1796 #PPD$S_STSST,- ; from response status
50 OD A2 04D0 1797 PPD$B_STATUS(R2),R0
04D3 1798
04D3 1799 $DISPATCH R0,TYPE=B,- ; Dispatch on subtype:
04D3 1800 <-
04D3 1801 <PPD$C_STPSV, RSP_PKTSTZ_VIO>,- ; Pkt size violation
04D3 1802 <PPD$C_STURP, RSP_UNREC_PKT>,- ; Unrecognized rec'd pkt
04D3 1803 <PPD$C_STINVD, RSP_CRASH_INVD>,- ; Invalid destination port
04D3 1804 <PPD$C_STURC, RSP_CRASH_URC>,- ; Unrecognized command
04D3 1805 <PPD$C_STABO, RSP_CRASH_ABO>,- ; Aborted command
04D3 1806 <PPD$C_OSEQ, RSP_OSEQ_ERR>,- ; Response had seq # mismatch
04D3 1807 <PPD$C_VCDCL, RSP_VCDCL_ERR>,- ; Seq msg rec'd on closed VCD
04D3 1808 >
04E5 1809
04E5 1810 UNIMP_STS_ERR:
04E5 1811
0124 31 04E5 1812 $DEBUGCHECK #ERR$V_DEB_UNSTS ; Optional bugcheck
04F8 1813 BRW RSP_CRASH_PORT ; Go init port crash
04FB 1814
04FB 1815 .DSABL LSB

```


- RSP_PATH_FAIL, PROCESS SINGLE PATH

```

04FB 1817      .SBTTL -      RSP_PATH_FAIL, PROCESS SINGLE PATH
04FB 1818      .SBTTL -      FAILURE
04FB 1819
04FB 1820      ;+
04FB 1821      ; RSP_PATH_FAIL records path failures in the configuration
04FB 1822      ; database, logs the error if necessary. If the received response
04FB 1823      ; was REQID, then there is a possibility that there are no
04FB 1824      ; good paths left. If no good paths remain, branch to RSP_CRASH_VC
04FB 1825      ; to crash the VC. If there is still a good path, return the response
04FB 1826      ; pkt to the msg/dg free queue if that is what the response bit in the
04FB 1827      ; locally executed command directed the port to do. I.e., if the
04FB 1828      ; response bit is 0, then, except for the path failure, the port
04FB 1829      ; would have put the command buffer on the free queue and this is
04FB 1830      ; what is reflected in the connection credit bookkeeping.
04FB 1831      ; Continue processing the response since it is ok except for the path failure.
04FB 1832
04FB 1833      ; An internal flag is set up on the stack during this routine. Normally, it
04FB 1834      ; is clear. It is set under the following conditions:
04FB 1835
04FB 1836      ; This path is bad AND the path was previously good AND
04FB 1837      ; the remaining path is good so the vc remains open.
04FB 1838
04FB 1839      ; When path status is all updated in the path block, check the internal
04FB 1840      ; flag. If clear, continue handling the response as described in the
04FB 1841      ; first paragraph. If the flag is set, and the response is a REQID,
04FB 1842      ; then use the response packet to send another REQID on the other path
04FB 1843      ; to test if it has also gone bad. The purpose of sending the extra
04FB 1844      ; REQID is to find out as quickly as possible if the VC is not working
04FB 1845      ; rather than waiting for the poller to do it.
04FB 1846
04FB 1847      ; Inputs:
04FB 1848
04FB 1849      ; R1      -PB addr
04FB 1850      ; R2      -Response addr
04FB 1851      ; R4      -PDT addr
04FB 1852
04FB 1853      ; Outputs:
04FB 1854
04FB 1855      ; R0,R1    -Destroyed
04FB 1856      ; Other registers -Preserved
04FB 1857      ; -
04FB 1858
04FB 1859      .ENABL  LSB
04FB 1860
04FB 1861      RSP_PATH_FAIL:
04FB 1862
04FB 1863      TSTL  R1      ; Check associated path block
04FB 1864      BEQL  30$    ; Branch if didn't get it
04FB 1865      CLRL  -(SP) ; Set flag on stack
04FB 1866      CLRL  R0     ; Assume path 0 bad
04FB 1867      BITB  #PPDSM_POSTS,- ; Any error code set in
04FB 1868      PPD$B_STATUS(R2) ; in path 0 status?
04FB 1869      BNEQ  10$    ; Branch if not
04FB 1870      INCL  R0     ; Else assume it's path 1 bad
04FB 1871
04FB 1872      10$: MOVAB  PB$B_PO_STS(R1)[R0],R0 ; Get addr of path status byte
04FB 1873      BLBC  (R0),20$ ; Branch if previous status bad

```



```
- FAILURE

FAEA' 30 0513 1874      BSBW  ELOG$PTH_ST_CHG      : Log failure of previously good path.
6E    D6 0516 1875      INCL  (SP)                  : Set flag to maybe send a REQID over
                                           : the remaining good path
60    01 8A 0518 1878 20$: BICB  #PBSM_CUR_PS,(R0)      : Set current path status bad
29    A1 89 051B 1879      BLSB3 PB$B_PO_STS(R1),-      : OR both path statuses
50    2A A1 051E 1880      : PB$B_P1_STS(R1),R0      : together
05    50 E8 0521 1881      BLBS  R0,DISPOSE_RSP        : Branch if one path still ok
8E    D5 0524 1882      TSTL  (SP)+                  : Remove flag from stack
0103  31 0526 1883      BRW   RSP_CRASH_VC            : Else crash VC
                                           :
                                           : DISPOSE_RSP:
                                           :
8E    D5 0529 1887      TSTL  (SP)+                  : Remove flag from stack
1E    13 052B 1888      BEQL  30$                     : Branch if flag clear
OE    A2 91 052D 1889      CMPB  PPD$B_OPC(R2),-      : Else was this a REQID that
05    05 0530 1890      : #PPD$C_REQID              : discovered the bad path?
18    12 0531 1891      BNEQ  30$                     : Branch if not
50    03 90 0533 1892      MOVB  #<<PPD$C_PSP0 @ PPD$V_PS>:PPD$M_RSP>,R0 : Assume we are going to turn
                                           : REQID around on path A, the
                                           : remaining good path
                                           : Path A still good?
29    A1 95 0536 1896      TSTB  PB$B_PO_STS(R1)      : Branch if so
03    12 0539 1897      BNEQ  25$                     : Branch if so
50    05 90 053B 1898      MOVB  #<<PPD$C_PSP1 @ PPD$V_PS>:PPD$M_RSP>,R0 : Else set up to poll path B
053E  1899
053E  1900
OF    A2 50 90 053E 1901 25$: MOVB  R0,PPD$B_FLAGS(R2) : Set flags in REQID pkt
OD    A2 94 0542 1902      CLRB  PPD$B_STATUS(R2)      : Clear status in REQID pkt
FB5F  30 0545 1903      BSBW  INT$INS_COMQL          : Requeue REQID to port
FD7E  31 0548 1904      BRW   REM_NEXT_RSP           : Go process next response
054B  1905
00    E0 054B 1906 30$: BBS   #PPD$V_RSP,-          : If response bit is set (response
06    OF A2 054D 1907      PPD$B_FLAGS(R2),40$      : requested), branch to process pkt
025D  30 0550 1908      BSBW  INT$INS_FREEQ          : Else insert pkt on appropriate
                                           : free queue
FD73  31 0553 1910      BRW   REM_NEXT_RSP           : and go for next response
0556  1911
FD77  31 0556 1912 40$: BRW   OPCODE_DISP            : Go process response
0559  1913
0559  1914      .DSABL  LSB
```


- RSP_UNREC_PKT, PROCESS RECEIPT OF

RSP_UNREC_PKT, PROCESS RECEIPT OF
UNRECOGNIZED PKT

```
0559 1916 .SBTTL -
0559 1917 .SBTTL -
0559 1918
0559 1919 :+
0559 1920 : RSP_UNREC_PKT logs the unrecognized packet, disables datagram receipt
0559 1921 : from the remote port, and discards the packet to the datagram free
0559 1922 : queue. Unrecognized packets with opcode of send maintenance reset/start
0559 1923 : are normal (the disk class driver sends them) and no action is taken
0559 1924 : on these.
0559 1925
0559 1926 : Inputs:
0559 1927
0559 1928 : R1 -PB addr
0559 1929 : R2 -Response addr
0559 1930 : R4 -PDT addr
0559 1931
0559 1932 : Outputs:
0559 1933
0559 1934 : R0-R3 -Destroyed
0559 1935 : Other registers -Preserved
0559 1936 :-
0559 1937
0559 1938 : .ENABL LSB
0559 1939
0559 1940 RSP_UNREC_PKT:
0559 1941
0559 1942 MOVZBL PPD$B_OPC(R2),R0 ; Extract the opcode
0559 1943 $DISPATCH R0,- ; If the opcode is
0559 1944 <<PPD$C_SNDST,20$>,- ; reset or
0559 1945 <PPD$C_SNDSTRT,20$>> ; start go return with no action
0565 1946
0565 1947 $DEBUGCHECK #ERR$V_DEB_URP ; Optional bugcheck
0578 1948 PUSHL R2 ; Save bad command addr
057A 1949 ASSUME PAER$K_ES_UPKT EQ 0
057A 1950 CLRL R0 ; Log unrecognized packet received
057C 1951 BSBW ELOG$PACKET1 ; error.
057F 1952 TSTL R1 ; Have we a VC open to this node?
0581 1953 BEQL 5$ ; Branch if not
0583 1954 BSBW ERR$CRASHVC ; Else crash the VC
0586 1955 BRB 10$ ; Dispose of dg
0588 1956
0588 1957 5$: BSBW INT$ALLOC_PPDDG ; Allocate a buffer to do SETCKT
058B 1958 BLBC R0,10$ ; Branch if none
058E 1959 MOVZBL PPD$B_PORT(R2),R0 ; Tell port to mark
0592 1960 BISL3 #<PPD$V_RSPa24>,- ; VC closed and to
059B 1961 <PPD$C_SETCKTa16>,R0,- ;
059B 1962 PPD$B_PORT(R2)
059B 1963 MOVZWL #PPD$M_DQI,PPD$W_MASK(R2) ; inhibit datagram reception
05A1 1964 MOVZWL #PPD$M_DQI,PPD$W_M_VAL(R2) ; from this remote port
05A7 1965 BBSS R0,PDT$B_DQIMAP(R4),8$
05AD 1966 8$: BSBW QHI
05B0 1967
05B0 1968 10$: POPL R2 ; Retrieve bad command addr
05B3 1969 BSBW INT$INS_DFREQ1 ; Return datagram entry to free queue
05B6 1970
05B6 1971 20$: BRW REM_NEXT_RSP ; Go for next response
05B9 1972
```

50 OE A2 9A

52 DD

50 D4

FA81' 30

51 D5

05 13

FA7A' 30

28 11

02BF 30

22 50 E9

OC A2 50 00190000 8F C9

10 A2 1000 8F 3C

14 A2 1000 8F 3C

00 0154 C4 50 E2

FA8F 30

52 8ED0

023E 30

FD10 31

PAINTR
V04-001

D 13

- UNRECOGNIZED PKT

05B9 1973

.DSABL LSB

16-SEP-1984 01:11:55
10-SEP-1984 01:15:57

VAX/VMS Macro V04-00
[DRIVER.SRC]PAINTR.MAR;2

Page 46
(28)

PAI
V04

- RSP_NO_PATH, PROCESS NO PATH

```
05B9 1975 .SBTTL - RSP_NO_PATH, PROCESS NO PATH
05B9 1976 .SBTTL - STATUS
05B9 1977 .SBTTL - RSP_PKTSIZ_VIO, PROCESS PACKET SIZE
05B9 1978 .SBTTL - VIOLATION STATUS
05B9 1979 .SBTTL - RSP_VC_CLOSED, PROCESS VC CLOSED
05B9 1980 .SBTTL - STATUS
05B9 1981
05B9 1982 :+
05B9 1983 : These three error status types require examination of the opcode and,
05B9 1984 : in some cases, the PPD type code to determine proper error handling.
05B9 1985 :
05B9 1986 : Each of the three entries pick up the address of the appropriate error action
05B9 1987 : table. Further processing consists of searching the opcodes in the action
05B9 1988 : table for a match. Failure to find a match results in a port crash since this
05B9 1989 : may be an error in software processing or a garbaged response. When the
05B9 1990 : correct opcode is located, then the PPD type is matched if required, and
05B9 1991 : the specified action routine is branched to.
05B9 1992 :
05B9 1993 : Register usage throughout is as follows:
05B9 1994 :
05B9 1995 : R2 -Response pkt addr
05B9 1996 : R3 -Addr of current PPD entry in action table
05B9 1997 : R4 -PDT addr
05B9 1998 : R5 -Addr of current opcode entry in action table
05B9 1999 :
05B9 2000 : Inputs:
05B9 2001 :
05B9 2002 : R1 -PB addr
05B9 2003 : R2 -Response pkt addr
05B9 2004 : R4 -PDT addr
05B9 2005 :
05B9 2006 : Outputs:
05B9 2007 :
05B9 2008 : R0-R2 -Destroyed
05B9 2009 : Other registers -Preserved
05B9 2010 :
05B9 2011 : Branch to REM_NEXT_RSP
05B9 2012 :-
05B9 2013 :
05B9 2014 : .ENABL LSB
05B9 2015 :
05B9 2016 RSP_NO_PATH:
05B9 2017
50 FE45 CF DE 05B9 2018 MOVAL ACT_NO_PATH,R0 ; Get no path action table
OC 11 05BE 2019 BRB 10$ ; Join common code
05C0 2020
05C0 2021 RSP_PKTSIZ_VIO:
05C0 2022
50 FE86 CF DE 05C0 2023 MOVAL ACT_PKTSIZ_VIO,R0 ; Get pkt size violation act table
05 11 05C5 2024 BRB 10$ ; Join common code
05C7 2025
05C7 2026 RSP_VC_CLOSED:
05C7 2027
50 FEBF CF DE 05C7 2028 MOVAL ACT_VC_CLOSED,R0 ; Get VC closed action table
55 50 D0 05CC 2029
05CF 2030 10$: MOVL R0,R5 ; Get action table addr
05CF 2031
```

- STATUS			
			05CF 2032 CMP_OPCODE:
			05CF 2033
	65 91		05CF 2034 CMPB EOASB_OPC(R5),-
	OE A2		05D1 2035 PPD\$B_OPC(R2)
	1E 12		05D3 2036 BNEQ NXT_OPCODE
53	02 A5	DE	05D5 2037 MOVAL EOASW_PPDTYP(R5),R3
			05D9 2038
			05D9 2039 CHK_ANY_PPD:
			05D9 2040
FFFF	8F 63	B1	05D9 2041 CMPW (R3), #-1
	0C 13		05DE 2042 BEQL CALL_ACTION
			05E0 2043
			05E0 2044 CMP_PPD:
			05E0 2045
12	A2 63	B1	05E0 2046 CMPW (R3), PPD\$W_MTYPE(R2)
	06 13		05E4 2047 BEQL CALL_ACTION
53	04 A3	DE	05E6 2048 MOVAL PPA\$C_LENGTH(R3),R3
	ED 11		05EA 2049 BRB CHK_ANY_PPD
			05EC 2050
			05EC 2051 CALL_ACTION:
			05EC 2052
50	02 A3	32	05EC 2053 CVTWL PPA\$W_RTN(R3),R0
	6340	17	05F0 2054 JMP (R3)[R0]
			05F3 2055
			05F3 2056
			05F3 2057 NXT_OPCODE:
			05F3 2058
50	01 A5	98	05F3 2059 CVTBL EOASB_NEXTOPC(R5),R0
	03 12		05F7 2060 BNEQ 20\$
	0023	31	05F9 2061 BRW RSP_CRASH_PORT
			05FC 2062
55	50 C0		05FC 2063 20\$: ADDL R0,R5
	CE 11		05FF 2064 BRB CMP_OPCODE
			0601 2065
			0601 2066 .DSABL LSB

- ACTION ROUTINES

```
0601 2068 .SBTTL - ACTION ROUTINES
0601 2069 .SBTTL - RSP_CACHECLR
0601 2070
0601 2071 :+
0601 2072 : RSP_CACHECLR calls the appropriate routine in PASCCTL to handle receipt
0601 2073 : of a cache clear message - a PPD sequenced message with PPD$W_MTYPE containing
0601 2074 : PPD$C_CACHECLR. This is a sequenced message sent out to a remote station
0601 2075 : associated with a closed (crashed) virtual circuit. It is sent after all
0601 2076 : CDT's associated with the closed VC have been DISCONNECTED and therefore
0601 2077 : signals that the port cache no longer holds any commands associated
0601 2078 : with that VC. The CACHECLR msg should always be received with fatal
0601 2079 : error status since it is sent over a closed VC, but this is not
0601 2080 : verified.
0601 2081
0601 2082 : If the earlier lookup of the path block failed, then SCSS$CACHECLR
0601 2083 : ends up crashing the port.
0601 2084
0601 2085 : Inputs:
0601 2086
0601 2087 : R1 -Addr of PB (0 if none)
0601 2088 : R2 -Addr of CACHECLR marker msg
0601 2089 : R4 -Addr of PDT
0601 2090
0601 2091 : Outputs:
0601 2092
0601 2093 : R0-R3 -Destroyed
0601 2094 : Other registers -Preserved
0601 2095 :-
0601 2096
0601 2097 .ENABL LSB
0601 2098
0601 2099 RSP_CACHECLR:
0601 2100
0601 2101 TSTL R1 ; Is there a PB?
0603 2102 BEQL 10$ ; Branch if not
0605 2103 MOVL R2,PB$L_CLSCKT_DG(R1) ; Else restore addr of cache
0609 2104 ; clear pkt to PB
0609 2105
0609 2106 10$: BSBW SCSS$CACHECLR ; Perform remaining circuit
060C 2107 ; cleanup (initiates a port
060C 2108 ; crash if no PB which will
060C 2109 ; take care of dangling SETCKT)
060C 2110 BRW REM_NEXT_RSP ; Get next response
060F 2111
060F 2112 .DSABL LSB
```

54 A1 51 D5 0601 2101 TSTL R1 ; Is there a PB?
04 13 0603 2102 BEQL 10\$; Branch if not
52 D0 0605 2103 MOVL R2,PB\$L_CLSCKT_DG(R1) ; Else restore addr of cache
; clear pkt to PB
F9F4' 30 0609 2106 10\$: BSBW SCSS\$CACHECLR ; Perform remaining circuit
; cleanup (initiates a port
; crash if no PB which will
; take care of dangling SETCKT)
; Get next response
FCBA 31 060C 2110 BRW REM_NEXT_RSP
060F 2111
060F 2112 .DSABL LSB

- RSP_CLOSED_VC

```

060F 2114      .SBTTL -          RSP_CLOSED_VC
060F 2115
060F 2116      ;+
060F 2117      ; RSP_CLOSED_VC handles errors that resulted in the port closing the
060F 2118      ; virtual circuit. It calls SCSS$VCCLOSED to notify SYSAP's
060F 2119      ; owning connections on this VC. It then branches to RSP_DRAIN_ERR
060F 2120      ; to dispose of the response containing error status.
060F 2121
060F 2122      Inputs:
060F 2123
060F 2124      R1                -PB addr
060F 2125      R2                -Response addr
060F 2126      R4                -PDT addr
060F 2127
060F 2128      Outputs:
060F 2129
060F 2130      R0-R3            -Destroyed
060F 2131      Other registers -Preserved
060F 2132      ;
060F 2133
060F 2134      .ENABL  LSB
060F 2135      RSP_CLOSED_VC:
060F 2136      MOVZBL #PAERSK_ES_PCVC, R0      ; Log port hardware has closed
060F 2137      BSBW  ELOG$PACKET1                ; the VC error.
060F 2138      MOVZWL #SS$VCCLOSED,R3            ; Assume aux status will be closed
060F 2139      ; due to other than host shutdown
060F 2140      BSBW  SCSS$VCCLOSED                ; Call VC closed handler
060F 2141      BRB   RSP_DRAIN_ERR                ; Go dispose of response
060F 2142      .DSABL  LSB

```

50 01 9A 060F 2136
F9EB' 30 0612 2137
53 21A4 8F 3C 0615 2138
F9E3' 30 061A 2139
18 11 061D 2141
061F 2142

- RSP_CRASH_PORT

```

061F 2144      .SBTTL -          RSP_CRASH_PORT
061F 2145
061F 2146      ;+
061F 2147      ; RSP_CRASH_PORT crashes the port by calling ERR$CRASH_PORT and
061F 2148      ; then draining the response currently held.
061F 2149      ;
061F 2150      Inputs:
061F 2151
061F 2152      R1                -PB addr (0 if none)
061F 2153      R2                -Response addr
061F 2154      R4                -PDT addr
061F 2155
061F 2156      Outputs:
061F 2157
061F 2158      R0-R2            -Destroyed
061F 2159
061F 2160      Other registers  -Preserved
061F 2161      :-
061F 2162
061F 2163      .ENABL LSB
061F 2164
061F 2165      INT$CRASH_PORT::
061F 2166      RSP_CRASH_PORT:
061F 2167
50    8002 8F    32 061F 2168      CVTWL  #<PAER$K_ES_CSHP!-
                                ^X8000>,R0      ; Log we are crashing the
0624 2169                                ; port error.
                                BSBW  ELOG$PACKET1 ; Force port to crash
0624 2170                                BSBW  ERR$CRASH_PORT ; Go dispose of response
0627 2171                                BRB   RSP_DRAIN_ERR
062A 2172
062C 2173      .DSABL LSB
062C 2174
062C 2175

```

```

- RSP_CRASH_VC
062C 2177 .SBTTL - RSP_CRASH_VC
062C 2178
062C 2179 :+
062C 2180 : RSP_CRASH_VC calls ERR$CRASHVC to initialize VC closure on an open VC
062C 2181 : (if there is one) and then branches to RSP_DRAIN_ERR to dispose of the error
062C 2182 : response. Return is taken from RSP_DRAIN_ERR.
062C 2183
062C 2184 : Inputs:
062C 2185 :
062C 2186 : R1 -PB addr (0 if none)
062C 2187 : R2 -Response addr
062C 2188 : R4 -PDT addr
062C 2189
062C 2190 : Outputs:
062C 2191 :
062C 2192 : R0-R2 -Destroyed
062C 2193 : Other registers -Preserved
062C 2194 :-
062C 2195
062C 2196 .ENABL LSB
062C 2197
062C 2198 RSP_CRASH_VC:
062C 2199
50 03 9A 062C 2200 MOVZBL #PAERSK_ES_SCVC, R0 ; Log a we are crashing the VC
F9CE' 30 062C 2201 BSBW ELOG$PACKET1 ; error.
F9CB' 30 0632 2202 BSBW ERR$CRASHVC ; Initiate crash (noop if no VC)
00 11 0635 2203 BRB RSP_DRAIN_ERR ; Go dispose of response
0637 2204
0637 2205 .DSABL LSB

```


- RSP_DRAIN_ERR

.SBTTL - RSP_DRAIN_ERR

```

0637 2207
0637 2208
0637 2209 :+
0637 2210 : RSP_DRAIN_ERR is called to dispose of a response once appropriate
0637 2211 : virtual circuit crash/sysap notification action has been taken.
0637 2212 : If the response bit is 0, then this packet is either a sent command
0637 2213 : intended to be returned to the free queue or a response occupying a
0637 2214 : free queue entry. The buffer is returned to the appropriate free queue.
0637 2215 : If the response bit is set, then this is a sent command with requested
0637 2216 : response regardless of success or failure. In this case, the buffer
0637 2217 : is returned to pool unless it is a SNDDG in which case it is processed
0637 2218 : as if no error occurred. Thus SNDDG's can be returned to the SYSAP if
0637 2219 : requested by the SYSAP.
0637 2220
0637 2221 : Inputs:
0637 2222
0637 2223 : R2 -Response addr
0637 2224 : R4 -PDT addr
0637 2225
0637 2226 : Outputs:
0637 2227
0637 2228 : R0-R2 -Destroyed
0637 2229 : Other registers -Preserved
0637 2230 :-
0637 2231
0637 2232 : .ENABL LSB
0637 2233
0637 2234 RSP_DRAIN_ERR:
0637 2235
06 00 E0 0637 2236 BBS #PPDSV RSP,- : If response bit set,
06 OF A2 0639 2237 PPD$B FLAGS(R2),20$ : go return to pool
0171 30 063C 2238 BSBW INT$INS_FREEQ : Put response back on appropriate
FC87 31 063F 2239 BRW REM_NEXT_RSP : free queue
0642 2240 : Go for next response
0E A2 91 0642 2241 20$: CMPB PPD$B OPC(R2),- : Is this a datagram?
01 0645 2243 #PPD$C SNDDG :
08 13 0646 2244 BEQL RSP_IGNORE_ERR : Branch if so to ignore error
0252 30 0648 2245 BSBW INT$DEAL_PRT : Deallocate msg/dg to pool
52 D4 064B 2246 CLRL R2 : Show pkt gone (debug aid)
FC79 31 064D 2247 BRW REM_NEXT_RSP : Go for next response
0650 2248
0650 2249 : .DSABL LSB

```

- RSP_IGNORE_ERR

```

0650 2251      .SBTTL -      RSP_IGNORE_ERR
0650 2252      .SBTTL -      RSP_DISCARD_ERR
0650 2253
0650 2254      :+
0650 2255      : RSP_IGNORE_ERR is branched to by the error action dispatcher. In this
0650 2256      : case, the error is ok and we want to clear the stack and go to the
0650 2257      : opcode dispatch to process the response as if there were no error.
0650 2258
0650 2259      : RSP_DISCARD_ERR is used for error responses that should be
0650 2260      : ignored and returned to pool without any processing.
0650 2261
0650 2262      : Inputs:
0650 2263
0650 2264      :      R2      -Response addr
0650 2265      :      R4      -PDT addr
0650 2266
0650 2267      :      (SP)    -Return to fork dispatcher
0650 2268
0650 2269      : Outputs:
0650 2270
0650 2271      :      none
0650 2272      :-
0650 2273
0650 2274      :      .ENABL  LSB
0650 2275
0650 2276      RSP_IGNORE_ERR:
0650 2277
FC7D  31 0650 2278      BRW      OPCODE_DISP      ; Go process response normally
0653 2279
0653 2280      RSP_DISCARD_ERR:
0653 2281
0247  30 0653 2282      BSBW     INT$DEAL_PKT      ; Deallocate msg or datagram
FC70  31 0656 2283      BRW      REM_NEXT_RSP      ; Go for next response
0659 2284
0659 2285      :      .DSABL  LSB

```


- OPTIONAL DEBUG BUGCHECKS

OPTIONAL DEBUG BUGCHECKS

```
0659 2287 .SBTTL -
0659 2288
0659 2289 :+
0659 2290 : These routines are dispatched to upon receipt of a response
0659 2291 : with error status that would normally result in crashing the
0659 2292 : port. Each entry here does a bugcheck or not depending upon
0659 2293 : the state of the flag in ERR$DEBUGCHECK for this type of error
0659 2294 : status.
0659 2295
0659 2296 : Inputs:
0659 2297
0659 2298 : R1 -PB address (0 if none)
0659 2299
0659 2300 : ERR$DEBUGCHECK -Longword of flags that
0659 2301 : enable/disable different types
0659 2302 : of $DEBUGCHECK.
0659 2303
0659 2304 : Outputs:
0659 2305
0659 2306 : All registers -Preserved
0659 2307 :-
0659 2308
0659 2309 .ENABL LSB
0659 2310
0659 2311 RSP_CRASH_NPUPD: ; No path + SNDMSG + unrecognized
0659 2312 ; PPD type
00AC 31 0659 2313 $DEBUGCHECK #ERR$V_DEB_NPUPD
066C 2314 BRW 10$
066F 2315
066F 2316 RSP_CRASH_PSV: ; Pkt size violation error
066F 2317
066F 2318 $DEBUGCHECK #ERR$V_DEB_PSV
0096 31 0682 2319 BRW 10$
0685 2320
0685 2321 RSP_CRASH_VCUPD: ; VC closed + SNDMSG + unrecognized
0685 2322 ; PPD type
0080 31 0685 2323 $DEBUGCHECK #ERR$V_DEB_VCUPD
0698 2324 BRW 10$
069B 2325
069B 2326 RSP_CRASH_INVBN: ; Invalid buffer name
069B 2327
069B 2328 $DEBUGCHECK #ERR$V_DEB_INVBN
006A 31 06AE 2329 BRW 10$
06B1 2330
06B1 2331 RSP_CRASH_BLV: ; Buffer length violation
06B1 2332
06B1 2333 $DEBUGCHECK #ERR$V_DEB_BLV
0054 31 06C4 2334 BRW 10$
06C7 2335
06C7 2336 RSP_CRASH_ACCV: ; Access violation during
06C7 2337 ; block xfer
3F 11 06C7 2338 $DEBUGCHECK #ERR$V_DEB_ACCV
06DA 2339 BRB 10$
06DC 2340
06DC 2341 RSP_CRASH_INVDP: ; Invalid destination port
06DC 2342
06DC 2343 $DEBUGCHECK #ERR$V_DEB_INVDP
```

- OPTIONAL DEBUG BUGCHECKS

2A	11	06EF	2344	BRB	10\$	
		06F1	2345			
		06F1	2346	RSP_CRASH_URC:		; Unrecognized command
		06F1	2347			
		06F1	2348	\$DEBUGCHECK #ERR\$V_DEB_URC		
15	11	0704	2349	BRB	10\$	
		0706	2350			
		0706	2351	RSP_CRASH_ABO:		; Aborted command
		0706	2352			
		0706	2353	\$DEBUGCHECK #ERR\$V_DEB_ABO		
00	11	0719	2354	BRB	10\$	
		071B	2355			
FF01	31	071B	2356	10\$: BRW	RSP_CRASH_PORT	; Come here if bugcheck not enabled
		071E	2357			; to go crash port and recover.
		071E	2358			
		071E	2359	RSP_OSEQ_ERR:		; Response with seq # mismatch --
		071E	2360			; either a legitimate duplicate
		071E	2361			; or a seq # error
		071E	2362	\$DEBUGCHECK #ERR\$V_DEB_OSEQ		
13	11	0731	2363	BRB	20\$; Action is force vc closure
		0733	2364			
		0733	2365	RSP_VCDCL_ERR:		; Seq msg rec'd on closed VCD
		0733	2366			
		0733	2367	\$DEBUGCHECK #ERR\$V_DEB_VCDCL		
		0746	2368			
FEE3	31	0746	2369	20\$: BRW	RSP_CRASH_VC	; Action is to force vc closure
		0749	2370			
		0749	2371	.DSABL	LSB	


```
- INTSFATALQ_RMFQ, ERROR REMOVING FROM M

F88D' 30 0770 2430 BSBW ERR$DISP_ENTRY ; Dispose of buffer.
50 8ED0 0773 2431 POPL R0 ; Restore which queue failed code.
OC 11 0776 2432 BRB 30$ ; Join remove queue processing.
0778 2433
0778 2434
0778 2435 INTSFATALQ_RSPQ:
0778 2436
50 02 9A 0778 2437 MOVZBL #PAER$K_ES_RQRM, R0 ; Get code showing which queue failed.
07 11 077B 2438 BRB 30$ ; Branch to common code.
077D 2439
077D 2440 INTSFATALQ_RDFQ:
077D 2441
50 01 9A 077D 2442 MOVZBL #PAER$K_ES_DQRM, R0 ; Get code showing which queue failed.
02 11 0780 2443 BRB 30$ ; Branch to common code.
0782 2444
0782 2445 INTSFATALQ_RMFQ:
0782 2446
50 D4 0782 2447 ASSUME PAER$K_ES_MQRM EQ 0
0784 2448 CLRL R0 ; Get code showing which queue failed.
0784 2449
0784 2450
50 80000000 8F C8 0784 2451 30$: $DEBUGCHECK #ERR$V_DEB_ILKQ ; Optionally, bugcheck on this error
F85F' 30 0797 2452 BISL #^X80000000, R0 ; Flag this as a crash the port error.
51 DD 079E 2453 BSBW ELOG$INTRLOCK ; Log the queue interlock error.
F85A' 30 07A1 2454 PUSHL R1 ; Save caller's R1
51 8ED0 07A3 2455 BSBW ERR$CRASH_PORT ; Init port crash
50 7F 8F 90 07A6 2456 POPL R1 ; Restore caller's R1
50 96 07A9 2457 MOVB #^X7F, R0 ; Clear Z bit
05 07AD 2458 INCB R0 ; and set overflow
07AF 2459 RSB ; Return to caller
07B0 2460
07B0 2461 .DSABL LSB
```


PACKET ALLOCATION/DEALLOCATION/DISPOSAL

```

0780 2463 .SBTTL PACKET ALLOCATION/DEALLOCATION/DISPOSAL ROUTINES
0780 2464 .SBTTL - INT$INS_FREEQ, DETERMINE IF PKT
0780 2465 .SBTTL - IS MSG OR DG AND
0780 2466 .SBTTL - INSERT OF FREE QUEUE
0780 2467
0780 2468 :+
0780 2469 : INT$INS_FREEQ examines the software structure type to determine if
0780 2470 : this packet is a CU message or datagram. It inserts messages on
0780 2471 : the message free queue and datagrams on the datagram free queue.
0780 2472 :
0780 2473 : Inputs:
0780 2474 :
0780 2475 : R2 -Packet addr
0780 2476 : R4 -PDT addr
0780 2477 :
0780 2478 : Outputs:
0780 2479 :
0780 2480 : R0 -Destroyed
0780 2481 : R2 -Zero (debug aid)
0780 2482 : Other registers -Preserved
0780 2483 :-
0780 2484
0780 2485 INT$INS_FREEQ:
0780 2486
0A A2 91 0780 2487 CMPB PPDSB_TYPE(R2),- ; Is this a
3B 0783 2488 #DYN$C_CIDG ; datagram?
3E 13 0784 2489 BEQL INT$INS_DFREQ1 ; Branch if so
05 11 0786 2490 BRB INT$INS_MFREQ1 ; Else do message

```

- INS_MFREEQ INSERT ON MESSAGE FREE QUEUE

```

07B8 2492      .SBTTL -      INS_MFREEQ      INSERT ON MESSAGE FREE QUEUE
07B8 2493
07B8 2494      :+
07B8 2495      : These routines insert a message buffer on the free queue. The
07B8 2496      : different entry points correspond to differing positions of R2
07B8 2497      : on entry. Specifically:
07B8 2498
07B8 2499      :      INS_MFREEQ1      R2 ->
07B8 2500      :
07B8 2501      :      PPD/SCS headr
07B8 2502      :
07B8 2503      :      INS_MFREEQ      R2 ->
07B8 2504      :
07B8 2505      :      SYSAP data
07B8 2506      :
07B8 2507      :
07B8 2508      :
07B8 2509      : Inputs:
07B8 2510      :
07B8 2511      :      R2      -Addr of message
07B8 2512      :      R3      -Addr of CDT
07B8 2513      :      R4      -Addr of PDT
07B8 2514      :
07B8 2515      : Outputs:
07B8 2516      :
07B8 2517      : -
07B8 2518
07B8 2519      INT$INS_MFREEQ::
07B8 2520
52  00B4 C4  C2  07B8 2521      SUBL2  PDT$L_MSGHDRSZ(R4),R2      ; Point to PPD start of buffer
07BD 2522
07BD 2523      INT$INS_MFREEQ1:
07BD 2524
07BD 2525      $INS_MFREEQ      : Insert on message free queue
52  D4  07BD 2526      CLRL  R2      ; Show packet gone
05  05  07DD 2527      RSB

```


- INS_DFREQ INSERT ON DATAGRAM FREE QUE

07DE 2529 .SBTTL - INS_DFREQ INSERT ON DATAGRAM FREE QUEUE

07DE 2530

07DE 2531

07DE 2532

07DE 2533

07DE 2534

07DE 2535

07DE 2536

07DE 2537

07DE 2538

07DE 2539

07DE 2540

07DE 2541

07DE 2542

07DE 2543

07DE 2544

07DE 2545

07DE 2546

07DE 2547

07DE 2548

07DE 2549

07DE 2550

07DE 2551

07DE 2552

07DE 2553

07DE 2554

07DE 2555

07DE 2556

07DE 2557

07DE 2558

07DE 2559

07DE 2560

07DE 2561

07DE 2562

```

52 0194 C4 C0 07DE 2563 ADDL PDT$L_DGNETHD(R4),R2 ; Step to start of PPD
    0194 C4 AE 07E3 2564 MNEGW PDT$L_DGNETHD(R4),- ; Put negative offset to default
      08 A2 07E7 2565 PPD$W_SIZE(R2) ; network header in size
      3B B0 07E9 2566 MOVW #DYN$C_CIDG,- ; Set structure type in
      0A A2 07EB 2567 PPD$B_TYPE(R2) ; PPD header
      05 11 07ED 2568 BRB INT$INS_DFREQ1
    07EF 2569
    07EF 2570 INT$INS_DFREQ::
52 0190 C4 C2 07EF 2571
    07F4 2572 SUBL2 PDT$L_DGHDRSZ(R4),R2 ; Point to PPD start of buffer
    07F4 2573
    07F4 2574 INT$INS_DFREQ1::
    07F4 2575
    52 D4 0812 2576 $INS_DFREQ ; Insert on datagram free queue
    05 0814 2577 CLRL R2 ; Show packet gone
    0814 2578 RSB

```

These routines insert a datagram buffer onto the free queue. The different entry points correspond to differing positions of R2 on entry. Specifically:

INS_DFREQX	R2 ->	----- DECnet header -----
INS_DFREQ1	R2 ->	----- PPD/SCS headr -----
INS_DFREQ	R2 ->	----- SYSAP data -----

Inputs:

R2 -Addr of datagram
R3 -Addr of CDT
R4 -Addr of PDT

Outputs:

R2 -Zeroed

INT\$INS_DFREQX::

INT\$INS_DFREQ::

INT\$INS_DFREQ1::

\$INS_DFREQ ; Insert on datagram free queue
CLRL R2 ; Show packet gone
RSB

- INT\$ALLOC_MSG, ALLOCATE A MSG BUFFER

```

0815 2580 .SBTTL - INT$ALLOC_MSG, ALLOCATE A MSG BUFFER FROM POOL
0815 2581 .SBTTL - INT$ALLOC_DG, ALLOCATE A DG BUFFER FROM POOL
0815 2582 .SBTTL - INT$ALLOC_DGPPD, ALLOCATE A BUFFER FOR
0815 2583 .SBTTL - PPD COMMAND
0815 2584 :+
0815 2585 : These routines allocate a single message or datagram buffer
0815 2586 : from nonpaged pool.
0815 2587 :
0815 2588 : Message format is simply the PPD/SCS header (PDT$L_MSGHDRSZ(R4))
0815 2589 : followed by application data (SCS$GW_MAXMSG bytes.)
0815 2590 :
0815 2591 : Datagram format is complex. It consists of a network header
0815 2592 : area used by the network SYSAP followed by the PPD/SCS header
0815 2593 : (PDT$L_DGHDRSZ(R4) bytes) followed by the application data.
0815 2594 : The network header size is, by default, the constant stored in
0815 2595 : PDT$L_DGNETHD. SYSAPs sending datagrams in buffers they
0815 2596 : allocate themselves may have a different size header provided
0815 2597 : they always request that the sent dg be returned to the SYSAP or
0815 2598 : pool. The sum of the default net header and the PPD/SCS header is
0815 2599 : stored in PDT$L_DGOVRHD(R4). Both the network overhead area
0815 2600 : and the PPD header begin with a standard VMS header including
0815 2601 : the structure type of DYN$C_CIDG. The structure size stored in
0815 2602 : the network header area is the total size of the buffer. The
0815 2603 : structure size stored in the PPD header is the negative offset from
0815 2604 : the start of the PPD header to the start of the network header.
0815 2605 :
0815 2606 : PPD datagram format is the same as SCS datagram, except that
0815 2607 : the space following the network header is large enough only
0815 2608 : for a CKTSET or smaller PPD message. The total size of the
0815 2609 : buffer allocated is: PDT$L_DGNETHD(R4) + PPD$W_LENGTH + 3*4.
0815 2610 : THE network header is unused, but is present in case of error
0815 2611 : recovery which returns the packet to pool via INT$DEAL_PKT/DG.
0815 2612 : PPD datagrams can be used only for commands issued with the response
0815 2613 : bit set. They must never be recycled to the free queue.
0815 2614 :
0815 2615 : Inputs:
0815 2616 :
0815 2617 : R4 -Addr of PDT
0815 2618 :
0815 2619 : Outputs:
0815 2620 :
0815 2621 : R0 -Status: LBC/LBS for fail/success
0815 2622 : R1 -Destroyed
0815 2623 : R2 -Addr of start of application data
0815 2624 : if status = success (MSG, DG, DGPPD)
0815 2625 : Addr of start of buffer (_DG1)
0815 2626 : other registers -Preserved
0815 2627 :
0815 2628 : PPD$B_TYPE(R2) -DYN$C_CIMSG/DYN$C_CIDG
0815 2629 : PPD$B_TYPE+1(R2) -0
0815 2630 : PPD$W_SIZE(R2) -Size of msg buffer structure, or, if dg,
0815 2631 : negative offset to start of net header =
0815 2632 : -(PDT$L_DGNETHD(R4))
0815 2633 :
0815 2634 :
0815 2635 : .ENABL LSB
0815 2636 :

```



```
- PPD COMMAND

0815 2637 INT$ALLOC_MSG::
0815 2638
50 00AC C4 DE 0815 2639 MOVAL PDT$L_WAITQFL(R4),R0 ; Get address of pool wait queue
50 60 D1 081A 2640 CMPL (R0),R0 ; Is the queue empty?
28 12 081D 2641 BNEQ 7$ ; No, make this CDRP wait too
53 DD 081F 2642 PUSHL R3 ; Save R3
51 00000000'GF 3C 0821 2643 MOVZWL G^SCS$GW_MAXMSG,R1 ; Get message size
51 00B4 C4 C0 0828 2644 ADDL PDT$L_MSGHDRSZ(R4),R1 ; including PPD/SCS header
00000000'GF 16 082D 2645 JSB G^EXE$ALONONPAGED ; Allocate the message
OD 50 E9 0833 2646 BLBC R0,5$ ; Branch if didn't get it
08 A2 51 B0 0836 2647 MOVW R1,PPD$W_SIZE(R2) ; Set structure size,
3C B0 083A 2648 MOVW #DYN$C_CIDG,- ; structure type and
0A A2 083C 2649 PPD$B_TYPE(R2) ; zero adjacent byte
52 00B4 C4 C0 083E 2650 ADDL PDT$L_MSGHDRSZ(R4),R2 ; Step to user portion
53 8ED0 0843 2651 5$: POPL R3 ; Restore R3
05 0846 2652 RSB ; Return
50 D4 0847 2653 7$: CLRL R0 ; Set failure status (low bit clear)
05 0849 2654 RSB
084A 2655
084A 2656 INT$ALLOC_PPDDG::
084A 2657
56 DD 084A 2659 PUSHL R6 ; Save R6
53 DD 084C 2660 PUSHL R3 ; Save R3
56 D4 084E 2661 CLRL R6 ; Add no SCS/PPD offset on exit
1C C1 0850 2662 ADDL3 #PPD$W_LENGTH+<3*4>,- ; Get size of net header
51 0194 C4 0852 2663 PDT$L_DGNETHD(R4),R1 ; + port header + small DG
1B 11 0856 2664 BRB 20$ ; Go allocate
0858 2665
0858 2666 INT$ALLOC_DG1::
0858 2667
56 DD 0858 2668 PUSHL R6 ; Save R6
56 D4 085A 2669 CLRL R6 ; Add no SCS/PPD offset on exit
07 11 085C 2670 BRB 10$
085E 2671
085E 2672 INT$ALLOC_DG::
085E 2673
56 DD 085E 2674 PUSHL R6 ; Save R6
56 0190 C4 D0 0860 2675 MOVL PDT$L_DGHDRSZ(R4),R6 ; Add no SCS/PPD offset on exit
53 DD 0865 2676
51 00000000'GF 3C 0865 2677 10$: PUSHL R3 ; Save R3
51 00B8 C4 C0 0867 2678 MOVZWL G^SCS$GW_MAXDG,R1 ; Get datagram size
086E 2679 ADDL PDT$L_DGOVRHD(R4),R1 ; including PPD/SCS header
0873 2680 ; and default net header
00000000'GF 16 0873 2681 20$: JSB G^EXE$ALONONPAGED ; Allocate the datagram
1A 50 E9 0879 2682 R0,30$ ; Branch if didn't get it
08 A2 51 B0 087C 2683 MOVW R1,PPD$W_SIZE(R2) ; Set structure size,
3B B0 0880 2684 MOVW #DYN$C_CIDG,- ; structure type and
0A A2 0882 2685 PPD$B_TYPE(R2) ; zero adjacent byte
52 0194 C4 C0 0884 2686 ; at buffer top
0194 C4 AE 0889 2687 ADDL PDT$L_DGNETHD(R4),R2 ; Step to addr of PPD header
08 A2 088D 2688 MNEGW PDT$L_DGNETHD(R4),- ; Put negative offset to start of
3B B0 088F 2689 PPD$W_SIZE(R2) ; net header in size field
0A A2 0891 2690 MOVW #DYN$C_CIDG,- ; structure in PPD header
52 56 C0 0893 2691 PPD$B_TYPE(R2)
2692 ADDL R6,R2 ; Add any extra offset
2693
```

PAINTR
V04-001

I 14

16-SEP-1984 01:11:55 VAX/VMS Macro V04-00
10-SEP-1984 01:15:57 [DRIVER.SRC]PAINTR.MAR;2

Page 64
(41)

```
- PPD COMMAND
53 8ED0 0896 2694 30$:
56 8ED0 0896 2695
    05 0899 2696
      089C 2697
      089D 2698
      089D 2699
      POPL R3
      POPL R6
      RSB
      .DSABL LSB
      ; Restore R3
      ; Restore R6
      ; Return
```


- INT\$DEAL_MSG, DEALLOCATE A MESSAGE BUF

```
089D 2701      .SBTTL -      INT$DEAL_MSG,  DEALLOCATE A MESSAGE BUFFER
089D 2702      .SBTTL -      INT$DEAL_DG,   DEALLOCATE A DATAGRAM BUFFER
089D 2703      .SBTTL -      INT$DEAL_PKT,  DEALLOCATE A DG OR MSG
089D 2704
089D 2705      :+
089D 2706      : INT$DEAL_MSG -- Given the address of the application data in a message
089D 2707      : buffer, deallocate the buffer to pool. Backs the pointer up from the
089D 2708      : start of the application data to the start of the PPD layer and depends
089D 2709      : upon PPD$W_SIZE being correctly set to the size of the buffer to deallocate.
089D 2710      :
089D 2711      : INT$DEAL_DG -- Given the address of the application data in a datagram
089D 2712      : buffer, deallocate the buffer to pool. Backs up the pointer from the
089D 2713      : start of the application data to the start of the PPD layer. Examines
089D 2714      : PPD$W_SIZE. If negative, uses as a negative offset to back the
089D 2715      : buffer pointer back up the start of the network header. If PPD$W_SIZE
089D 2716      : is positive, then there is no net header, so join common deallocation.
089D 2717      :
089D 2718      : INT$DEAL_DG1 -- Given the start of the PPD layer, check for network
089D 2719      : header as described above and deallocate the dg.
089D 2720      :
089D 2721      : INT$DEAL_PKT -- Given the address of the PPD layer, check if the
089D 2722      : packet is of type CIDG. If so, go to INT$DEAL_DG1. Else join
089D 2723      : message deallocation code.
089D 2724      :
089D 2725      : Inputs:
089D 2726      :
089D 2727      :      R2      -Addr of PPD layer (_DG1 or _PKT)
089D 2728      :      R4      Addr of application data (_DG, _MSG)
089D 2729      :      R4      -Addr of PDT (_MSG, _DG)
089D 2730      :
089D 2731      : Outputs:
089D 2732      :
089D 2733      :      R0,R2      -Destroyed
089D 2734      :
089D 2735      :      Other registers      -Preserved
089D 2736      :
089D 2737      :
089D 2738      : .ENABL  LSB
089D 2739      :
089D 2740      INT$DEAL_PKT::
089D 2741      :
089D 2742      CMPB  PPD$B_TYPE(R2),-      : Is this a datagram?
089D 2743      #DYN$C_CIDG      :
089D 2744      BNEQ  10$      : Branch if not
089D 2745      BRB   INT$DEAL_DG1      : Else join deallocate of dg
089D 2746      : given PPD header address
089D 2747      :
089D 2748      INT$DEAL_DG::
089D 2749      :
089D 2750      SUBL  PDT$L_DGHDRSZ(R4),R2      : Back up to PPD header
089D 2751      :
089D 2752      INT$DEAL_DG1::
089D 2753      :
089D 2754      CVTWL  PPD$W_SIZE(R2),R0      : Get net header size indicator
089D 2755      BGEQ  10$      : Branch if no net header
089D 2756      MOVAB  (R2)[R0],R0      : Back up to start of net header
089D 2757      BRB   20$      : Join common deallocate
```

0A A2 91
3B
18 12
05 11

52 0190 C4 C2

50 08 A2 32
0B 18
50 6240 9E
08 11

			08B6	2758			
			08B6	2759	INT\$DEAL_MSG::		
			08B6	2760			
52	00B4	C4	C2	08B6	2761	SUBL2	PDT\$L_MSGHDRSZ(R4),R2 ; Set addr of buffer
				08BB	2762		
	50	52	D0	08BB	2763	10\$: MOVL	R2,R0 ; Transfer register
				08BE	2764		
				08BE	2765	20\$: PUSH	#^M<R1,R3> ; Save registers
00000000	0A	BB	08BE	2765		JSB	G^EXE\$DEANONPAGED ; Deallocate nonpaged pool
	GF	16	08C0	2766		POPR	#^M<R1,R3> ; Restore registers
	0A	BA	08C6	2767		RSB	; Return
		05	08C8	2768			

- DFQ2POOL REMOVE FROM DATAGRAM FREE QUE

```

08C9 2770      .SBTTL -      DFQ2POOL      REMOVE FROM DATAGRAM FREE QUEUE
08C9 2771
08C9 2772      :+
08C9 2773      :
08C9 2774      : Inputs:
08C9 2775      :
08C9 2776      :      R3      -Addr of CDT
08C9 2777      :      R4      -Addr of PDT
08C9 2778      :
08C9 2779      : Outputs:
08C9 2780      :
08C9 2781      :      R2      -Addr of datagram
08C9 2782      :-
08C9 2783
08C9 2784      INT$DFQ2POOL::
08C9 2785
08C9 2786      $REM_DFREQ      ; Remove from datagram
CB  1C 08DD 2787      BVC      INT$DEAL_DG1      ; Good, get rid of it
05 08DF 2788      RSB      ; Fail, leave with status

```

- MFQ2POOL REMOVE FROM MESSAGE FREE QUEUE
.SBTTL - MFQ2POOL REMOVE FROM MESSAGE FREE QUEUE

```

08E0 2790
08E0 2791
08E0 2792 :+
08E0 2793 :
08E0 2794 : Inputs:
08E0 2795 :
08E0 2796 : R3
08E0 2797 : R4
08E0 2798 :
08E0 2799 : Outputs:
08E0 2800 :
08E0 2801 : R2
08E0 2802 :-
08E0 2803 :
08E0 2804 INT$MFQ2POOL::
08E0 2805
08E0 2806 $REM_MFREEQ ; Remove from mesage free queue
C5 1C 08F4 2807 BVC 10$ ; Good, get rid of it
05 08F6 2808 RSB ; Failed, return status
08F7 2809
08F7 2810 .DSABL LSB
08F7 2811
08F7 2812
08F7 2813
08F7 2814
08F7 2815 .END

```


NAME	VALUE	UNIT	TYPE
ERRSV DEB VCUPD	*****	X	01
EXESAICONONPAGED	*****	X	01
EXESDEANONPAGED	*****	X	01
EXESGL LOCKRTRY	*****	X	01
EXESIOFORK	*****	X	01
FATAL CNFERR	= 001EE700		
FPC\$REC_CNFREC	*****	X	01
FPC\$REC_DGREC	*****	X	01
FPC\$REC_MSGREC	*****	X	01
FPC\$REC_RDCNT	*****	X	01
FPC\$REC_SNDDG	*****	X	01
FPC\$REC_SNDMSG	*****	X	01
HANDLE INT	0000029C	R	01
HDWR ERR CODE	00000228	R	01
IDB\$C_CSR	= 00000000		
IDB\$C_UCBLST	= 00000018		
INIS\$FORK	*****	X	01
INIS\$PORT	*****	X	01
INT\$ALLOC_DG	0000085E	RG	01
INT\$ALLOC_DG1	00000858	RG	01
INT\$ALLOC_MSG	00000815	RG	01
INT\$ALLOC_PPDDG	0000084A	RG	01
INT\$CLRCACHE	0000013A	RG	01
INT\$CRASH_PORT	0000061F	RG	01
INT\$DEAL_DG	000008A5	RG	01
INT\$DEAL_DG1	000008AA	RG	01
INT\$DEAL_MSG	000008B6	RG	01
INT\$DEAL_PKT	0000089D	RG	01
INT\$DFQ2POOL	000008C9	RG	01
INT\$DISP_SENDDG	000003D1	RG	01
INT\$FATALQ_CQH	00000758	R	01
INT\$FATALQ_CQL	00000753	R	01
INT\$FATALQ_IDFQ	00000749	R	01
INT\$FATALQ_IMFQ	0000074E	R	01
INT\$FATALQ_RDFQ	0000077D	R	01
INT\$FATALQ_RMFQ	00000782	R	01
INT\$FATALQ_RSPQ	00000778	R	01
INT\$INS_COMQH	0000003F	RG	01
INT\$INS_COMQL	000000A7	RG	01
INT\$INS_DFREQ	000007EF	RG	01
INT\$INS_DFREQ1	000007F4	RG	01
INT\$INS_DFREQX	000007DE	RG	01
INT\$INS_FREEQ	000007B0	R	01
INT\$INS_MFREEQ	000007B8	RG	01
INT\$INS_MFREEQ1	000007BD	R	01
INT\$MFQ2POOL	000008E0	RG	01
INT\$MRESET	000000EC	RG	01
INT\$MSTART	00000111	RG	01
INT\$READCNT	000000C5	RG	01
INT\$REQDAT	00000087	RG	01
INT\$SNDDAT	00000098	RG	01
INT\$SNDDG	00000006	RG	01
INT\$SNDDG1	0000000F	RG	01
INT\$SNDMSG	00000028	RG	01
INT\$SNDMSG1	0000005D	RG	01
INT\$TRNMSG	00000076	RG	01
INV_OPCODE	00000345	R	01

PAINTR
Symbol table

B 15

16-SEP-1984 01:11:55 VAX/VMS Macro V04-00
10-SEP-1984 01:15:57 [DRIVER.SRC]PAINTR.MAR;2

Page 70
(44)

NDTS_CI	= 00000038		
NO_CI750	00000213	R	01
NO_RSP	000002D9	R	01
NXT_OPCODE	000005F3	R	01
OPCODE_DISP	000002D0	R	01
OTHER_CNF_ERR	000001FE	R	01
PASINT	00000160	RG	01
PAERSK_ES_CSHP	= 00000002		
PAERSK_ES_DQIN	= 00000006		
PAERSK_ES_DQRM	= 00000001		
PAERSK_ES_HCIN	= 00000003		
PAERSK_ES_HWER	= 00000002		
PAERSK_ES_LCIN	= 00000004		
PAERSK_ES_LST0	= 00000003		
PAERSK_ES_LST1	= 00000009		
PAERSK_ES_LST2	= 00000007		
PAERSK_ES_LST3	= 00000009		
PAERSK_ES_LST4	= 0000000C		
PAERSK_ES_MQIN	= 00000005		
PAERSK_ES_MQRM	= 00000000		
PAERSK_ES_PCV	= 00000001		
PAERSK_ES_PDWN	= 00000003		
PAERSK_ES_PUP	= 00000004		
PAERSK_ES_RQRM	= 00000002		
PAERSK_ES_SCVC	= 00000003		
PAERSK_ES_UPKT	= 00000000		
PAERSK_ES_UXIN	= 00000005		
PAERSK_ET_DALT	= 00000003		
PAERSK_ET_LMLT	= 00000042		
PA_CNF	00000000		
PA_CNF_M_CRD	= 00010000		
PA_CNF_V_NOCI	= 0000000C		
PA_CNF_V_PDN	= 00000017		
PA_CNF_V_PUP	= 00000016		
PA_CQ0	00000908		
PA_CQ0_M_CQC	= 00000001		
PA_CQ1	0000090C		
PA_CQ1_M_CQC	= 00000001		
PA_CQ2	00000910		
PA_CQ3	00000914		
PA_DFQ	00000928		
PA_DFQ_M_DFQC	= 00000001		
PA_MADR	00000014		
PA_MDATR	00000018		
PA_MFQ	0000092C		
PA_MFQ_M_MFQC	= 00000001		
PA_MTC	00000930		
PA_MTEC	00000934		
PA_PDC	00000920		
PA_PEC	0000091C		
PA_PESR	0000093C		
PA_PFAR	00000938		
PA_PIC	00000924		
PA_PMC	00000004		
PA_PMC_M_MIE	= 00000004		
PA_PMC_M_MIN	= 00000001		
PA_PPR	00000940		

PA_PQBRR	00000904
PA_PS	00000900
PA_PSR	00000918
PA_PSR_M_PSC	= 00000001
PA_PS_M_MTE	= 80000000
PA_PS_M_RQA	= 00000001
PA_PS_V_MFQE	= 00000001
PBSB_P0_STS	= 00000029
PBSB_P1_STS	= 0000002A
PBSB_RSTATION	= 0000000C
PBSC_LENGTH	= 00000054
PBSC_PALENGTH	00000060
PBSL_CLSCKT_DG	00000054
PBSM_CUR_PS	= 00000001
PDTB_DQIMAP	00000154
PDTB_HSHUT_DG	000001B0
PDTB_MAX_PORT	0000017C
PDTB_NXT_PORT	0000017E
PDTB_P0_CBSTS	00000180
PDTB_P1_LBSTS	00000181
PDTB_PLUGMAP	00000134
PDTB_PORTMAP	00000114
PDTB_PORT_NUM	0000017D
PDTB_REQIDPS	0000017F
PDTSC_LENGTH	= 000000E4
PDTSC_PAREGBASE	000000E4
PDTSC_PAREGEND	00000110
PDTSC_PQB	= 000001E0
PDTSL_CNF	000000E4
PDTSL_CQ0	000000F0
PDTSL_CQ1	000000F4
PDTSL_DFQ	000000FC
PDTSL_DFQHDR	00000208
PDTSL_DGHDRSZ	00000190
PDTSL_DGNETHD	00000194
PDTSL_DGOVRHD	= 000000B8
PDTSL_DQELOGOUT	000002E0
PDTSL_GPTBASE	0000022C
PDTSL_GPTLEN	00000230
PDTSL_LBDG	00000184
PDTSL_MFQ	00000100
PDTSL_MFQHDR	0000020C
PDTSL_MQELOGOUT	00000320
PDTSL_MSGHDRSZ	= 000000B4
PDTSL_MTC	00000104
PDTSL_P FAR	00000108
PDTSL_PMC	000000E8
PDTSL_POLLERDUE	0000018C
PDTSL_POOLDUE	00000188
PDTSL_PPR	0000010C
PDTSL_PS	000000EC
PDTSL_PSR	000000F8
PDTSL_SPTBASE	00000224
PDTSL_SPTLEN	00000228
PDTSL_VBDT	0000021C
PDTSL_VPQB	00000218
PDTSL_WAITQFL	= 000000AC

PAINTR
Symbol table

C 15

16-SEP-1984 01:11:55 VAX/VMS Macro V04-00
10-SEP-1984 01:15:57 [DRIVER.SRC]PAINTR.MAR;2

Page 71
(44)

PDT\$M_PUP = 00000002
PDT\$Q_COMQ2 = 000001F0
PDT\$Q_COMQ3 = 000001F8
PDT\$Q_COMQBASE = 000001E0
PDT\$Q_COMQH = 000001E8
PDT\$Q_COMQL = 000001E0
PDT\$Q_DFREQ = 000001D0
PDT\$Q_FORMPB = 00000174
PDT\$Q_MFREQ = 000001D8
PDT\$Q_RSPQ = 00000200
PDT\$Q_TEMP_RSPQ = 0000019C
PDT\$V_PWF_CLNUP = 00000000
PDT\$W_BDT\$EN = 00000220
PDT\$W_DQ\$EN = 00000210
PDT\$W_LP\$RT_STS = 00000110
PDT\$W_MQ\$EN = 00000214
PDT\$W_PBCOUNT = 00000112
PDT\$W_STDGDYN = 00000198
PDT\$W_STDGUSED = 0000019A
PORT_ERR = 000000E6 R 01
PPA\$C_LENGTH = 00000004
PPA\$W_PPD\$TYP = 00000000
PPA\$W_RT\$N = 00000002
PPD\$B_DEF_ST = 0000001C
PPD\$B_FLAGS = 0000000F
PPD\$B_HWVERS = 00000034
PPD\$B_LB\$DATA = 00000012
PPD\$B_LCB_O = 00000012
PPD\$B_LCB_LP\$RT = 00000010
PPD\$B_LCB_NP\$RT = 0000000F
PPD\$B_LCB_OP\$C = 00000011
PPD\$B_LCB_P\$RT = 0000000E
PPD\$B_OP\$C = 0000000E
PPD\$B_P\$RT = 0000000C
PPD\$B_P\$RT\$OCOL = 0000001A
PPD\$B_R\$STATE = 00000025
PPD\$B_R\$T_P\$RT = 00000024
PPD\$B_S\$TATUS = 0000000D
PPD\$B_SW\$FLAG = 0000000B
PPD\$B_S\$YSTEMID = 00000014
PPD\$B_T\$YPE = 0000000A
PPD\$C_CACHE\$CLR = 00008000
PPD\$C_CACHE\$LEN = 00000002
PPD\$C_CN\$FREC = 00000023
PPD\$C_D\$ATREC = 00000031
PPD\$C_DG\$REC = 00000021
PPD\$C_ID\$REC = 0000002B
PPD\$C_IN\$VTC = 00000018
PPD\$C_LB\$REC = 0000002D
PPD\$C_LB\$LENGTH = 00000046
PPD\$C_LCB_\$DATA = 00000013
PPD\$C_L\$ENGTH = 00000012
PPD\$C_MC\$NFREC = 00000029
PPD\$C_MD\$ATREC = 00000033
PPD\$C_MIN_DGS\$IZ = 00000050
PPD\$C_MS\$GREC = 00000022
PPD\$C_O\$SEQ = 00000005

PPD\$C_P\$SPO = 00000001
PPD\$C_P\$SP1 = 00000002
PPD\$C_R\$DCNT = 0000001A
PPD\$C_R\$EQDAT = 00000008
PPD\$C_R\$EQID = 00000005
PPD\$C_R\$EQMDAT = 0000000E
PPD\$C_R\$ETCNF = 00000003
PPD\$C_R\$ETDAT = 00000011
PPD\$C_S\$CS_DG = 00000003
PPD\$C_S\$CS_MS\$G = 00000004
PPD\$C_S\$ET\$CKT = 00000019
PPD\$C_S\$NDDAT = 00000010
PPD\$C_S\$NDDG = 00000001
PPD\$C_S\$NDLB = 0000000D
PPD\$C_S\$NDMDAT = 00000012
PPD\$C_S\$NDMSG = 00000002
PPD\$C_S\$NDR\$T = 00000006
PPD\$C_S\$ND\$STRT = 00000007
PPD\$C_S\$TABO = 00000004
PPD\$C_S\$TIN\$VDP = 00000002
PPD\$C_S\$TP\$SV = 00000000
PPD\$C_S\$T\$URC = 00000003
PPD\$C_S\$T\$URP = 00000001
PPD\$C_T\$YPACCV = 00000004
PPD\$C_T\$YPBLV = 00000003
PPD\$C_T\$YB\$MSE = 00000006
PPD\$C_T\$YPI\$NVBN = 00000002
PPD\$C_T\$Y\$NP = 00000005
PPD\$C_T\$Y\$OK = 00000000
PPD\$C_T\$Y\$OT\$HER = 00000007
PPD\$C_T\$Y\$VCC = 00000001
PPD\$C_V\$CDCL = 00000006
PPD\$K_LB_\$LENGTH = 00000046
PPD\$K_L\$ENGTH = 00000012
PPD\$L_B\$LINK = 00000004
PPD\$L_DG_\$DISC = 00000028
PPD\$L_F\$LINK = 00000000
PPD\$L_IN_\$VCD = 00000018
PPD\$L_LB\$CRC = 00000042
PPD\$L_PO_\$ACK = 00000010
PPD\$L_PO_\$NAK = 00000014
PPD\$L_PO_\$NR\$SP = 00000018
PPD\$L_P1_\$ACK = 0000001C
PPD\$L_P1_\$NAK = 00000020
PPD\$L_P1_\$NR\$SP = 00000024
PPD\$L_REC_\$BOFF = 00000028
PPD\$L_REC_\$NAME = 00000024
PPD\$L_RP\$RT_FC\$N = 00000020
PPD\$L_RP\$RT_\$REV = 0000001C
PPD\$L_RP\$RT_\$TYP = 00000018
PPD\$L_S\$ND_\$BOFF = 00000020
PPD\$L_S\$ND_\$NAME = 0000001C
PPD\$L_S\$T_\$ADDR = 00000018
PPD\$L_X\$CT_\$LEN = 00000018
PPD\$M_D\$ISPOSE = 00000001
PPD\$M_DQ\$I = 00001000
PPD\$M_P\$OSTS = 00000006

PAINTR
Symbol table

D 15

16-SEP-1984 01:11:55 VAX/VMS Macro V04-00
10-SEP-1984 01:15:57 [DRIVER.SRC]PAINTR.MAR;2

Page 72
(44)

PPDSM_RSP	= 00000001		
PPDSQ_CURTIME	00000048		
PPDSQ_NODENAME	00000040		
PPDSQ_SWINCARN	00000028		
PPDSQ_XCT_ID	00000010		
PPDSS_STSST	= 00000004		
PPDSS_STSTYP	= 00000003		
PPDST_HWTYPE	00000030		
PPDST_SWTYPE	00000020		
PPDST_SWVERS	00000024		
PPDSV_DISPOSE	= 00000000		
PPDSV_ERR	= 00000000		
PPDSV_PS	= 00000001		
PPDSV_RSP	= 00000000		
PPDSV_STSST	= 00000001		
PPDSV_STSTYP	= 00000005		
PPDSW_LCB_LEN7	0000000C		
PPDSW_LENGTH	00000010		
PPDSW_MASK	00000010		
PPDSW_MAXDG	0000001C		
PPDSW_MAXMSG	0000001E		
PPDSW_MTYPE	00000012		
PPDSW_M_VAL	00000014		
PPDSW_SIZE	00000008		
PSR_ERRORS	= 0000007E		
PS_ERR	000001A7	R	01
PWR_DN	00000244	R	01
PWR_UP	00000267	R	01
QHI	0000003F	R	01
QLOW	000000A7	R	01
REC_CNFRFC	0000035D	R	01
REC_DATREC	0000035D	R	01
REC_DGREC	00000368	R	01
REC_IDREC	0000037F	R	01
REC_INVTC	000003FC	R	01
REC_LBREC	00000394	R	01
REC_MCNFRFC	00000345	R	01
REC_MDATREC	00000345	R	01
REC_MSGREC	0000039A	R	01
REC_RDCNT	000003A5	R	01
REC_REQDAT	00000345	R	01
REC_REQID	000003FC	R	01
REC_REQMDAT	00000345	R	01
REC_RETCNF	00000345	R	01
REC_RETDAT	00000345	R	01
REC_SETCKT	000003AB	R	01
REC_SNDAT	00000345	R	01
REC_SNDG	000003CC	R	01
REC_SNDLB	00000345	R	01
REC_SNDMDAT	00000345	R	01
REC_SNDMSG	000003F1	R	01
REC_SNDRST	000003FC	R	01
REC_SNDSTRT	000003FC	R	01
REINIT_PORT	00000220	R	01
REM_NEXT_RSP	000002C9	R	01
REM_RSP	000002AB	R	01
RSP_CACHECLR	00000601	R	01

RSP_CLOSED_VC	0000060F	R	01
RSP_CRASH_ABO	00000706	R	01
RSP_CRASH_ACCV	000006C7	R	01
RSP_CRASH_BLV	000006B1	R	01
RSP_CRASH_INVBN	0000069B	R	01
RSP_CRASH_INVDP	000006DC	R	01
RSP_CRASH_NPUPD	00000659	R	01
RSP_CRASH_PORT	0000061F	R	01
RSP_CRASH_PSV	0000066F	R	01
RSP_CRASH_URC	000006F1	R	01
RSP_CRASH_VC	0000062C	R	01
RSP_CRASH_VCUPD	00000685	R	01
RSP_DISCARD_ERR	00000653	R	01
RSP_DRAIN_ERR	00000637	R	01
RSP_ERROR	000004B0	R	01
RSP_IGNORE_ERR	00000650	R	01
RSP_NO_PATH	000005B9	R	01
RSP_OSEQ_ERR	0000071E	R	01
RSP_PATH_FAIL	000004FB	R	01
RSP_PKTSTZ_VIO	000005C0	R	01
RSP_SUBTYP_CHK	000004CD	R	01
RSP_UNREC_PKT	00000559	R	01
RSP_VCDCL_ERR	00000733	R	01
RSP_VC_CLOSED	000005C7	R	01
SBIERR	= FC000000		
SCSSCACHECLR	*****	X	01
SCSSGW_MAXDG	*****	X	01
SCSSGW_MAXMSG	*****	X	01
SCSSSETCKT_CLSD	*****	X	01
SCSSVCCLOSED	*****	X	01
SIZ...	= 00000001		
SS\$ABORT	= 0000002C		
SS\$CTRLERR	= 00000054		
SS\$NORMAL	= 00000001		
SS\$NOSUCHNODE	= 0000028C		
SS\$POWERFAIL	= 00000364		
SS\$VCCLOSED	= 000021A4		
SYSAP\$C_DGREC	= 00000000		
SYSAP\$C_DISPP0	= 00000002		
SYSAP\$C_DISPQ	= 00000000		
SYSAP\$C_DISPRT	= 00000001		
UCB\$B_ERTCNT	= 00000080		
UCB\$B_LMERTCNT	000000D2		
UCB\$B_LMERTMAX	000000D3		
UCB\$B_LMEST	000000D0		
UCB\$B_LMET	000000D1		
UCB\$K_ERRDGBYTS	= 000000B4		
UCB\$K_LMPKTBYTS	= 00000040		
UCB\$L_CICMD	000000F0		
UCB\$L_DPC	= 0000009C		
UCB\$L_MSGFKBLK	= 000000A0		
UCB\$L_PDT	= 00000084		
UCB\$N_LSADDR	000000D8		
UCB\$N_LSID	000000DE		
UCB\$N_RSADDR	000000E4		
UCB\$N_RSID	000000EA		
UCB\$T_MSGDATA	000000F8		

PAINTR
Symbol table

E 15

16-SEP-1984 01:11:55 VAX/VMS Macro V04-00
10-SEP-1984 01:15:57 [DRIVER.SRC]PAINTR.MAR;2

Page 73
(44)

UCBST_OPAO_TEMP	=	000000B8		
UCBSW_DEVSTS	=	00000068		
UCBSW_LMERRCNT		000000D4		
UCBSW_MSGBYTCNT		000000F4		
UCBSW_MSGPPDTYP		000000F6		
UCB_M_FKLOCK	=	00000002		
UCB_V_FKLOCK	=	00000001		
UNIMP_STS_ERR		000004E5	R	01
UNRECOV_ERR		00000238	R	01

+-----+
! Psect synopsis !
+-----+

PSECT name	Allocation	PSECT No.	Attributes														
ABS	00000000 (0.)	00 (0.)	NOPIC	USR	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC	BYTE				
\$\$\$115_DRIVER	000008F7 (2295.)	01 (1.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	LONG				
\$AB\$\$	00000944 (2372.)	02 (2.)	NOPIC	USR	CON	ABS	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE				

+-----+
! Performance indicators !
+-----+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	38	00:00:00.04	00:00:01.18
Command processing	137	00:00:00.42	00:00:03.43
Pass 1	577	00:00:18.74	00:01:01.95
Symbol table sort	9	00:00:01.89	00:00:07.77
Pass 2	401	00:00:05.15	00:00:20.20
Symbol table output	1	00:00:00.23	00:00:00.98
Psect synopsis output	0	00:00:00.01	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1165	00:00:26.49	00:01:35.54

The working set limit was 2550 pages.
154099 bytes (301 pages) of virtual memory were used to buffer the intermediate code.
There were 100 pages of symbol table space allocated to hold 1749 non-local and 92 local symbols.
2815 source lines were read in Pass 1, producing 27 object records in Pass 2.
50 pages of virtual memory were used to define 46 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name	Macros defined
-\$255\$DUA28:[DRIVER.OBJ]PALIB.MLB;1	17
-\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	9
-\$255\$DUA28:[SYSLIB]STARLET.MLB;2	8
TOTALS (all libraries)	34

2053 GETS were required to define 34 macros.

There were no errors, warnings or information messages.

PAINTR
VAX-11 Macro Run Statistics

F 15

16-SEP-1984 01:11:55 VAX/VMS Macro V04-00
10-SEP-1984 01:15:57 [DRIVER.SRC]PAINTR.MAR;2

Page 74
(44)

MACRO/LIS=LISS:PAINTR/OBJ=OBJ\$:PAINTR MSRC\$:PAINTR/UPDATE=(ENH\$:PAINTR)+EXECML\$/LIB+LIB\$:PALIB.MLB/LIB

0114 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

PAINT
LIS

PAFPCALL
LIS

PAINT
LIS

PAMONIT
LIS